Sampling Is Not Selecting: Verifiability as the Bottleneck of Inference-Time Scaling

Murilo Salem[1], **Luísa Böhm**[1], **Henrique dos Reis**[1], **Lúcio Vagner Carvalho**[1], **Theo Ferrugem**[1], **Anderson Ferrugem**[1]

[1] Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico (UFPel)
Pelotas – RS – Brasil

{mcsalem, lcbohm, hdreis, lvcarvalho, tdferrugem, ferrugem}@inf.ufpel.edu.br

Abstract. *Inference-time scaling can improve language model performance without changing model weights, but extra compute is not a task-independent resource: its value depends on both candidate generation and candidate selection. We evaluate greedy decoding, self-consistency, reference-answer oracle selection, and self- refinement under matched maximum generation-token allowances on Qwen2.5 instruction models across mathematical and general-reasoning benchmarks. The original best-of-N procedure scores candidates against the benchmark reference answer during selection; we therefore report it as Oracle@N, an upper bound on candidate coverage rather than a verifier available at deployment. On GSM8K with Qwen2.5-1.5B, Oracle@N rises from 52.9% at $N = 1$ to 96.4% at $N = 64$, and on MATH from 23.4% to 59.6%. Self-consistency saturates earlier, and on BBH it falls from 30.8% to 28.8% over the same range. Self-refinement also degrades on GSM8K, from 45.9% at one step to 33.7% at eight steps in the tested configuration. Extra samples create opportunity, but converting candidate coverage into deployed accuracy requires a selection mechanism that can be evaluated independently of the benchmark answer.*

1. Introduction

Large language models can spend compute at inference time in ways that go beyond a single forward generation. Chain-of-thought prompting encourages models to expose intermediate reasoning steps (Wei et al. 2022); self-consistency samples multiple reasoning traces and returns the most common final answer (Wang et al. 2022); verifier-guided decoding generates many candidates and selects the one judged most likely to be correct (Cobbe et al. 2021); and iterative refinement asks the model to critique and revise its own previous output (Madaan et al. 2023). These methods are appealing because they improve the deployment-time behavior of a fixed model. They shift some of the burden from training scale to inference-time computation.

Yet the phrase "inference-time scaling" can hide an important ambiguity. Extra compute can be used to produce more independent samples, longer deliberations, explicit critiques, or adaptive searches. These choices are not equivalent. More samples help only if at least one sample is correct and the procedure can select it. More refinement helps only if the model can diagnose its own failure modes. Majority voting helps only if correctness is correlated with answer frequency. The empirical question, therefore, is not

simply whether more inference compute helps, but under what task and verifier conditions it is converted into accuracy.

This paper argues that *verifiability* is a primary bottleneck for inference-time scaling. We use the term verifiability operationally: a task is more verifiable when candidate outputs can be scored by a reliable, low-noise procedure at test time. GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021) are relatively verifiable because final answers can be extracted and checked by exact matching, numeric normalization, or symbolic equivalence. BBH (Suzgun et al. 2022) is less verifiable in our setup because many tasks require semantic answer comparison, and our available judging signal is substantially weaker. Code tasks such as HumanEval (Chen et al. 2021) and MBPP (Austin et al. 2021) should in principle be highly verifiable through execution, but our current execution-based verifier produced invalid zero-score runs and is therefore excluded from the main empirical claims.

We study this question under matched maximum generation-token allowances. For a baseline per-example output-token cap B and multiplier N, parallel sampling methods generate N candidates with cap B, while self-refinement makes N sequential revisions with the same cap. This accounting makes the allowed output-token budget explicit, but it does not equalize actual generated tokens, input and revision-context tokens, verifier cost, latency, or FLOPs. Crucially, the reference-answer selection used in our best-of-N runs has access to the benchmark answer y; its results are therefore Oracle@N coverage measurements, not the performance of a deployable verifier.

The empirical pattern is sharp. On mathematical benchmarks, best-of-N scales strongly with N and dominates self-consistency at high budgets. On GSM8K, best-of-N improves by 43.5 percentage points from $N = 1$ to $N = 64$; self-consistency improves by 28.3 points and saturates near 83%. On MATH, best-of-N improves by 36.2 points, while self-consistency improves by 14.6 points. On BBH, however, additional samples do not produce the same effect: greedy decoding reaches 50.0%, best-of-N reaches 49.6% at $N = 64$, and self-consistency remains well below greedy. Self-refinement provides a second negative case: for the 1.5B model on GSM8K, additional refinement steps reduce accuracy rather than improving it.

The contributions of this paper are:

1. We separate candidate coverage from deployable candidate selection in an inference-time sampling study.
2. We report the reference-answer selection results as Oracle@N, making their oracle access and upper-bound interpretation explicit.
3. We show that self-consistency can saturate or degrade as sampling grows, and that the tested self-refinement configuration can degrade a small model's answers.
4. We provide a conservative evaluation rule: sampling results should report both how candidates are generated and how the final candidate is selected.

2. Related Work

Our work builds on a line of research that improves language model performance by changing how a fixed model is used at inference time rather than how it is trained. The starting point is chain-of-thought prompting, which showed that models solve more complex arithmetic, commonsense, and symbolic reasoning problems when prompted to generate

intermediate reasoning steps before committing to a final answer (Wei et al. 2022). This finding motivated a broader family of methods that leave the model weights untouched and instead modify the decoding procedure. We are closest to this line, but our concern is not whether rationales help relative to direct answering; rather, we ask how different uses of the same extra inference budget behave under matched compute.

Within this family, the methods differ mainly in how they convert additional samples into a final answer. Self-consistency replaces greedy decoding with multiple sampled reasoning paths and returns the most frequent final answer (Wang et al. 2022). Its central assumption is that correct answers recur across diverse traces more often than any individual incorrect one. This can be effective when a task has a unique short answer, but majority vote is not verification: it is a distributional signal from the generator itself, whereas verification is an external or semi-external correctness check. A central observation of this paper is that the distinction is not merely conceptual—self-consistency improves on GSM8K yet saturates below best-of-N, and it performs poorly on BBH in our setting.

Verifier-guided selection makes that correctness check explicit, and it has long been attractive for mathematical reasoning because final answers, and sometimes intermediate steps, can be checked independently of the generator. GSM8K was in fact introduced together with verifier-based methods that rank generated solutions at test time (Cobbe et al. 2021), and more recent work on test-time compute emphasizes search against verifier or reward models as one of the principal mechanisms by which inference compute improves performance (Snell et al. 2024). Our experiments rely on simpler task-specific verifiers rather than learned process reward models, but they isolate the same principle: generating many candidates helps only when selection among them is reliable.

Repeated sampling work also distinguishes coverage from selection. Brown et al. show that the fraction of problems solved by at least one sample can scale strongly, while majority vote and reward-model selection need not convert that coverage into the same deployed performance when automatic verification is unavailable (Brown et al. 2024). Our Oracle@N measurements follow this distinction: they quantify coverage under benchmark-answer access and do not evaluate a selector available at deployment.

A different way to spend inference compute is to revise rather than resample. Self-Refine proposes an iterative loop in which the model produces an initial answer, generates feedback on it, and refines it without additional supervised training (Madaan et al. 2023). This is appealing precisely because it needs no separate verifier, but it shifts the burden onto the base model, which must now detect and repair its own mistakes. Our results expose a failure mode of that assumption: with Qwen2.5-1.5B on GSM8K, refinement depth correlates negatively with accuracy, suggesting that useful self-correction requires sufficient model capacity or a stronger external feedback signal.

We study these strategies on three benchmarks chosen for their differing verification properties: GSM8K, a grade-school math word-problem benchmark (Cobbe et al. 2021); MATH, a competition-level mathematical reasoning benchmark (Hendrycks et al. 2021); and BBH, a collection of challenging BIG-Bench reasoning tasks (Suzgun et al. 2022). The base models are drawn from the Qwen2.5 instruction-tuned family (Qwen Team 2025). We also attempted the code benchmarks HumanEval and MBPP, but treat them only as excluded tasks because our local code-verification pi-

peline did not produce valid measurements.

3. Method

3.1. Problem Setting

Let x be an input problem and let y be the reference answer. A generation strategy produces one or more candidate outputs $o_1, \ldots, o_N$. Each candidate contains free-form text and an extracted final answer $a_i = e(o_i)$, where e is an answer-extraction procedure. A selection rule s returns the final answer reported by the system. Accuracy is measured as

$$\mathrm{Acc} = \frac{1}{M} \sum_{j=1}^{M} I\left[\hat{a}_j = y_j\right], \tag{1}$$

where M is the number of examples and equality is implemented by the task-specific scorer.

The central experimental variable is the compute multiplier N. We interpret N as the number of candidate samples for parallel strategies and as the number of refinement steps for self-refinement. Since strategies allocate compute differently, we use an iso-compute token budget to make comparisons more meaningful.

3.2. Matched Maximum Generation-Token Allowance

For each dataset, we define a per-generation maximum output-token cap B: 512 tokens for GSM8K, 1024 for MATH, and 256 for BBH. At multiplier N, self- consistency and Oracle@N generate N candidates, each capped at B tokens; self-refinement makes N sequential revisions with the same per-step cap. Thus, the maximum allowed output tokens scale as NB.

Greedy decoding is deterministic and reported at $N = 1$. Self-consistency samples with temperature 0.7 and returns the majority extracted answer. Oracle@N samples with temperature 0.8. Self-refinement uses temperature 0.7. This is token-cap accounting rather than a strict compute comparison: we do not measure actual generated tokens, context length, verifier cost, latency, or FLOPs.

3.3. Reference-Answer Selection and Evaluation

For mathematical tasks, the evaluator extracts final answers using GSM8K-style markers, boxed LaTeX expressions, or numeric fallbacks, then checks exact match, symbolic equivalence through SymPy, or numerical tolerance. In the original best-of-N implementation, this evaluator scores each candidate against the benchmark reference answer y *before* selection. The method selects the first candidate with maximal binary score; ties among equally scored candidates are therefore resolved by generation order. We call this Oracle@N because y is unavailable during real deployment. It measures whether a sampled set contains an answer accepted by the benchmark evaluator.

For BBH, the analogous score uses normalized exact and substring matching against the reference answer; it is likewise an oracle measurement and is not a semantic judge. HumanEval and MBPP remain excluded. The HumanEval harness executed generated completions without prepending the prompt's function signature and imports, so the all-zero code results are invalid rather than evidence about code generation.

3.4. Hypothesis

Our hypothesis is that the marginal value of inference compute depends on verifier reliability:

$$\Delta\mathrm{Acc}(N) \propto q(v), \tag{2}$$

where $q(v)$ denotes the quality of the verification or selection signal. This expression is not intended as a fitted statistical model. It states the qualitative prediction tested here: when the verifier can reliably distinguish correct from incorrect candidates, increasing N should improve accuracy; when the selection signal is weak, increasing N should saturate or degrade.

4. Experimental Setup

4.1. Models

The primary model is Qwen2.5-1.5B-Instruct. This model has the most complete experimental coverage across strategies, datasets, and budgets. We also include partial Qwen2.5-7B-Instruct results where available, mainly to check whether the qualitative best-of-N versus self-consistency pattern persists at a larger scale. The 7B runs are not complete enough to support the same full comparison as the 1.5B runs.

4.2. Datasets

The main experiments use three datasets. GSM8K contains 1319 grade-school math word problems in the evaluated split. MATH is evaluated on 500 competition-style mathematics problems. BBH is evaluated on 250 examples from challenging reasoning tasks. These datasets were selected because they vary in answer-verification properties: GSM8K and MATH have relatively clear final-answer checking, while BBH has a weaker answer-comparison signal in our setup.

4.3. Budgets and Temperatures

We evaluate budgets $N \in \{1, 2, 4, 8, 16, 32, 64\}$ where runs are available. The main table reports $N \in \{1, 4, 16, 64\}$ for readability. Greedy decoding uses temperature 0.0. Self-consistency uses temperature 0.7. Oracle@N uses temperature 0.8. Self-refinement uses temperature 0.7. The differing temperatures are a limitation of the original runs: the curves jointly vary candidate generation and selection. We therefore avoid treating their difference as a controlled causal estimate of selection alone. A controlled comparison would reuse one candidate pool for both selectors or generate both methods with identical decoding parameters.

4.4. Metrics

The primary metric is exact task accuracy after the strategy-specific selection step. We also report a simple scaling-efficiency diagnostic:

$$E = \frac{\mathrm{Acc}_{N=64} - \mathrm{Acc}_{N=1}}{\log_2 64}. \tag{3}$$

This quantity measures accuracy gain per doubling of the sampling budget. It is useful for comparing scaling slopes, but it should not be over-interpreted because it compresses non-linear curves into one number.

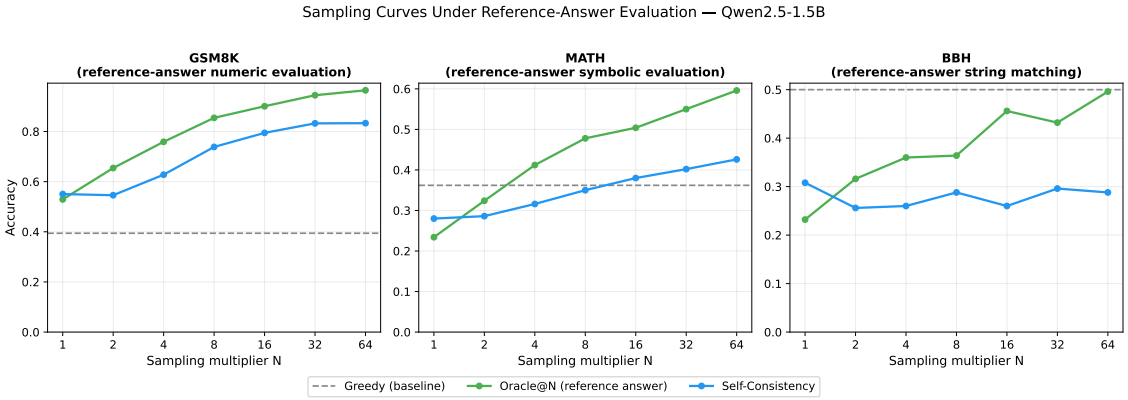


Figura 1. Inference-time scaling by task verifiability for Qwen2.5-1.5B-Instruct. GSM8K and MATH admit relatively reliable final-answer verification, and best-of-N converts additional samples into large accuracy gains. BBH uses a weaker answer-comparison signal in this setup, and additional samples do not yield the same improvement.

5. Results

5.1. Scaling Separates by Verifiability

The central finding of this paper is that inference-time scaling does not improve accuracy uniformly: whether additional samples help is governed by how reliably the system can select among them, not by how many it draws. Figure 1 establishes this by tracking self-consistency and best-of-N against a greedy reference line on three tasks that differ in verifiability, all with Qwen2.5-1.5B-Instruct.

On the two tasks with reliable final-answer checking, both strategies improve, but verifier-guided selection scales far more steeply. On GSM8K, best-of-N climbs 43.5 points, from 52.9% at $N = 1$ to 96.4% at $N = 64$, whereas self-consistency rises only from 55.0% to 83.3% and visibly flattens at high budgets—leaving a 13.1-point gap in best-of-N's favor. MATH reproduces the pattern under a harder distribution: best-of-N gains 36.2 points (23.4% to 59.6%) against just 14.6 for self-consistency (28.0% to 42.6%).

BBH breaks this pattern, and does so informatively. Here the answer-comparison signal is weak, and neither sampling strategy can turn extra candidates into accuracy. Greedy decoding alone reaches 50.0%; best-of-N recovers from a low 23.2% at $N = 1$ to 49.6% at $N = 64$, essentially climbing back to—but never past—the greedy line, while self-consistency drifts *downward*, from 30.8% to 28.8%. This is the sharpest negative case in our study: large amounts of additional sampling compute fail to help, and can mildly hurt, once selection is unreliable.

Because Oracle@N uses y during selection, these gains quantify candidate coverage under the benchmark evaluator; they do not demonstrate that an independently available verifier would obtain the same accuracy at deployment.

The scaling-efficiency diagnostic E tells the same story in one number. Best-of-N reaches 0.073 on GSM8K and 0.060 on MATH, comfortably above the 0.047 and 0.024 of self-consistency on the same tasks. On BBH the diagnostic turns slightly negative for

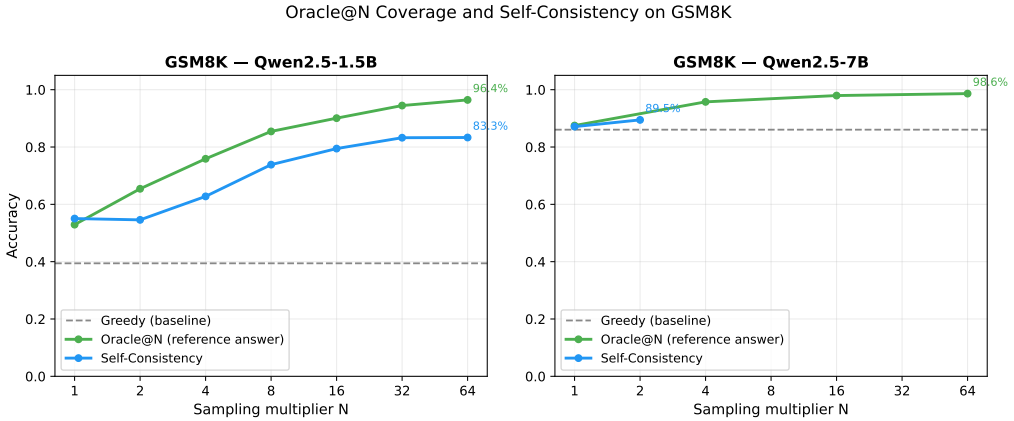


Figura 2. Best-of-N and self-consistency on GSM8K. Both methods use multiple samples, but they use different selection signals. Self-consistency selects the most frequent extracted answer; best-of-N selects the candidate that passes a task verifier. Under equal sampling budgets, verifier-guided selection continues improving after majority voting saturates.

self-consistency; best-of-N stays positive but only recovers toward greedy rather than surpassing it, echoing the curves rather than adding a new claim.

5.2. Best-of-N and Self-Consistency Are Mechanistically Different

If extra samples were the whole story, self-consistency and Oracle@N would scale together. Figure 2 shows that their observed curves diverge on GSM8K. Both methods draw multiple samples from the same base model, but the original curves use different temperatures (0.7 for self-consistency and 0.8 for Oracle@N). They are therefore descriptive rather than a fully controlled estimate of selection alone. In addition, Oracle@N uses the reference answer during selection and should be interpreted as an upper bound on candidate coverage rather than as a deployable verifier.

At small budgets the two methods track each other, since both mainly benefit from exploring several reasoning paths. They diverge once the budget is large enough that voting saturates while verification does not. Self-consistency reaches 79.5% at $N = 16$ and then barely moves—83.2% at $N = 32$, 83.3% at $N = 64$—whereas best-of-N keeps climbing over the same range, from 90.1% to 93.3% to 96.4%.

The two methods use different selection signals. Self-consistency asks which answer the model emits most often. Oracle@N asks whether a sampled candidate is accepted by a reference-answer evaluator. The latter can identify correct candidates only because it has access to y. Thus, the observed gap diagnoses headroom in the sampled candidate set, but does not establish that an independently available verifier would recover the same gap at deployment. This is why folding both methods under a single "sample-and-select" label obscures the mechanism that actually matters.

The partial 7B runs point the same way. The larger model starts from a much higher greedy accuracy, yet best-of-N still adds headroom, reaching 98.6% at the largest available budget. Self-consistency coverage at 7B is too sparse for a full comparison, so we read this only as corroborating evidence that verifier-guided selection keeps paying off as model size grows.

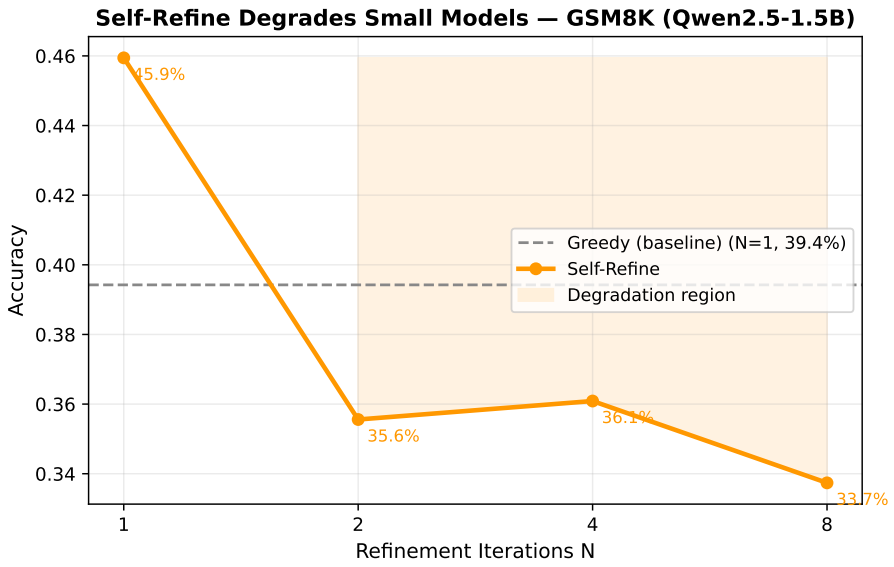


Figura 3. Self-refinement on GSM8K with Qwen2.5-1.5B-Instruct. Accuracy decreases as the number of refinement iterations increases, suggesting that iterative self-correction requires sufficient model capacity or a stronger feedback signal.

5.3. Self-Refinement Can Degrade Small Models

Sampling in parallel is not the only way to spend a token budget; self-refinement spends it sequentially, revising one answer instead of drawing many. Figure 3 shows that for Qwen2.5-1.5B on GSM8K this backfires.

Accuracy decreases overall, but not monotonically at every reported budget: it is 45.9% at $N = 1$, 35.6% at $N = 2$, 36.1% at $N = 4$, and 33.7% at $N = 8$. Thus, the tested configuration performs worse after additional refinement, although the present evidence does not isolate whether the cause is model size, prompting, temperature, or the revision procedure. The result should therefore be interpreted as a failure mode of this self-refinement configuration rather than evidence for a general capacity threshold.

Accuracy decreases overall, but not monotonically at every reported budget: it is 45.9% at $N = 1$, 35.6% at $N = 2$, 36.1% at $N = 4$, and 33.7% at $N = 8$. Thus, the tested configuration performs worse after additional refinement, although the present evidence does not isolate whether the cause is model size, prompting, temperature, or the revision procedure.

The failure is distinct in kind from BBH. On BBH the missing ingredient is a reliable *external* check that can choose among parallel candidates; in self-refinement it is a reliable *internal* check that can tell an improvement from a regression. Different signal, same lesson: an extra inference step helps only when something can tell whether it actually improved the answer.

5.4. Main Accuracy Table

Table 1 collects the Qwen2.5-1.5B numbers at representative budgets, underpinning the curves in Figures 1 and 3.

Tabela 1. Accuracy (%) by strategy and compute budget N (Qwen2.5-1.5B-Instruct). --- = no run at that budget.

Dataset	Strategy	$N = 1$	$N = 4$	$N = 16$	$N = 64$
4*GSM8K	Greedy	39.4	—	—	—
	Self-Consistency	55.0	62.8	79.5	83.3
	Best-of-N	52.9	75.9	90.1	96.4
	Self-Refine	45.9	36.1	—	—
4*MATH	Greedy	36.2	—	—	—
	Self-Consistency	28.0	31.6	38.0	42.6
	Best-of-N	23.4	41.2	50.4	59.6
	Self-Refine	—	—	—	—
4*BBH	Greedy	50.0	—	—	—
	Self-Consistency	30.8	26.0	26.0	28.8
	Best-of-N	23.2	36.0	45.6	49.6
	Self-Refine	—	—	—	—

Read across tasks, the table separates into three regimes. On GSM8K both sampling strategies far exceed their $N = 1$ baselines, but best-of-N pulls clear at large budgets. On MATH the same ordering holds with an even steeper best-of-N slope. On BBH the ranking inverts: greedy is the strongest or tied-strongest reported condition despite spending a fraction of the inference compute of $N = 64$ sampling. The deciding factor across all three is not how much was sampled but whether a trustworthy verifier was available to exploit it.

5.5. Iso-Compute Snapshot at a Moderate Budget

Finally, Figure 4 provides a mixed-budget descriptive snapshot. Greedy appears at its deterministic $N = 1$ baseline, while the remaining strategies are shown at $N = 4$. Because the methods are shown at different values of N, this figure is not an iso-compute comparison.

At this budget, Oracle@N is higher than self-consistency on GSM8K and MATH under reference-answer selection. On BBH, greedy remains competitive. This compact view is descriptive only and should be read together with the selection and temperature limitations discussed above.

Even at this modest budget the task-dependence is already visible, well before the large-N regime. Best-of-N is comfortably ahead on GSM8K and again leads self-consistency on MATH, while on BBH greedy stays competitive because the extra samples arrive without a selection signal strong enough to use them. The qualitative split this paper argues for is therefore not an artifact of pushing N to its extreme—it is present as soon as sampling begins.

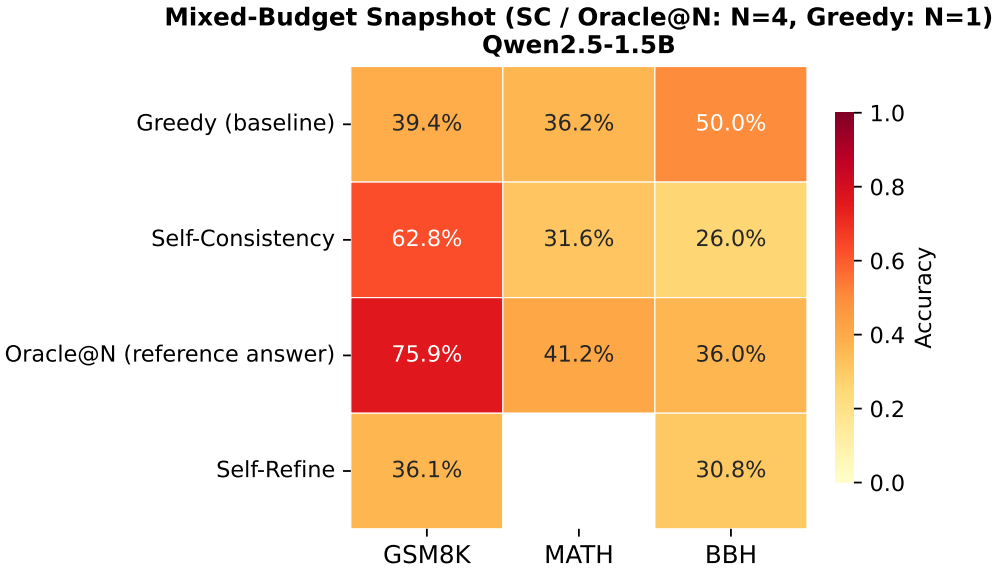


Figura 4. Mixed-budget snapshot. Greedy is shown at $N = 1$, while self-consistency, Oracle@N, and self-refinement are shown at $N = 4$. Oracle@N uses reference-answer selection. This figure is a descriptive summary, not an iso-compute comparison.

6. Discussion

6.1. Why Verifiability Matters

Inference-time scaling has two separable components: candidate generation and candidate selection. Increasing N can increase the chance that at least one sampled output is correct. Final deployed accuracy, however, also requires selecting that candidate without access to the benchmark answer.

Our Oracle@N curves diagnose the first component under reference-answer access. They should not be read as evidence that a practical verifier is available. Self-consistency is a deployable aggregation rule, but its comparison with the original Oracle@N curves is confounded by temperature. The central implication is therefore evaluative: a sampling study should distinguish coverage from the accuracy of the selector that will actually be used at deployment.

The same reasoning explains why BBH does not show the same scaling profile. In our setup, BBH scoring is closer to normalized answer matching than robust semantic verification. Additional samples increase answer diversity, but the selection mechanism cannot reliably exploit that diversity. The result is not that BBH can never benefit from inference-time compute. Rather, the result is that BBH needs a stronger verifier, task-specific checker, or judge before large sampling budgets can be expected to pay off.

6.2. A Practical Strategy Rule

These results do not prescribe a deployment policy because no deployable verifier is evaluated. Oracle@N measures candidate coverage with access to benchmark answers, while self-consistency is an aggregation rule that does not use those answers. Future work should evaluate candidate generation and selection separately: when a task has a genuine

automatic checker, the checker itself should be measured for reliability, cost, and failure modes.

The evidence supports a narrower evaluation recommendation. Inference-time sampling studies should report how candidates are generated, what information the selector receives, and whether the selector is available at deployment. This distinction prevents oracle coverage from being interpreted as deployed system accuracy.

6.3. Implications for Evaluation

These findings affect how inference-time methods should be evaluated. Aggregate results across heterogeneous tasks can obscure whether gains arise from candidate generation or selection; evaluations should therefore report the selection regime as well as benchmark accuracy. Likewise, token-budget accounting makes generation allowances explicit, but it does not equalize actual token use, context cost, verifier cost, latency, or FLOPs.

7. Limitations

The strongest claims rest on Qwen2.5-1.5B-Instruct, which has the most complete coverage; the 7B results are uneven, with a full best-of-N curve on GSM8K but only small-N runs for self-consistency and MATH, so we read them as corroborating rather than primary evidence. Larger models, including the planned 32B and reasoning-specialized variants, are still needed to test whether these failure modes persist at higher capacity. Coverage is also incomplete by task: HumanEval and MBPP are excluded because the local code-verification pipeline returned invalid all-zero results. This is a measurement failure, not evidence that code resists inference-time scaling—if anything, execution makes code a natural domain for strong verification, and it should be revisited with a validated sandbox.

The original runs use one model family, one fully covered model size, representative rather than matched temperatures, and one seed. We report no multi-seed variance. Bootstrap intervals over examples may characterize dataset-sampling uncertainty, but they would not replace repeated generations with independent seeds. Actual token use, input costs, verifier costs, latency, and FLOPs are also unreported.

8. Conclusion

Inference-time scaling is not a single accuracy knob but two separable steps: generating candidate solutions and selecting among them. Adding compute reliably helps the first; whether it helps the second—and therefore whether accuracy improves at all—depends on the quality of the selection signal. Our experiments make this split concrete. Where final answers can be checked, verifier-guided best-of-N converts samples into large gains (52.9% to 96.4% on GSM8K), while self-consistency saturates earlier because answer frequency is a weaker proxy than verification. Where the signal is weak, the same compute stops paying off: extra sampling fails to beat greedy on BBH, and self-refinement actively degrades a small model, showing that internal critique is capacity-limited in the same way external selection is verifier-limited.

The practical lesson is not that the reported Oracle@N procedure can be deployed, but that repeated sampling may create substantial candidate coverage. Turning that coverage into system accuracy requires a selector that does not use benchmark answers and is

evaluated for reliability, cost, and failure modes. Reporting how a method selects, what information the selector receives, and how reliable that selection is should be as standard as reporting how much it samples.

Referências

[Austin et al. 2021] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. (2021). Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

[Brown et al. 2024] Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. (2024). Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

[Chen et al. 2021] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

[Cobbe et al. 2021] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

[Hendrycks et al. 2021] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

[Madaan et al. 2023] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

[Qwen Team 2025] Qwen Team (2025). Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

[Snell et al. 2024] Snell, C., Lee, J., Xu, K., and Kumar, A. (2024). Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

[Suzgun et al. 2022] Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. (2022). Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

[Wang et al. 2022] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

[Wei et al. 2022] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.