

The Art of Scaling Test-Time Compute for Large Language Models

Aradhye Agarwal^{m,ω*} Ayan Sengupta^ω Tanmoy Chakraborty^ω

^mMicrosoft Research

^ωIndian Institute of Technology Delhi

Abstract

Test-time scaling (TTS)—the dynamic allocation of compute during inference—is a promising direction for improving reasoning in large language models (LLMs). However, a systematic comparison of well-known TTS strategies under identical conditions is missing, and the influence of model type and problem difficulty on performance remains unclear. To address these gaps, we conduct the first large-scale study of TTS, spanning over thirty billion tokens generated using eight open-source LLMs (7B to 235B parameters), across four reasoning datasets. We observe three consistent trends: (1) no single TTS strategy universally dominates; (2) reasoning models exhibit distinct trace-quality patterns across problem difficulty and trace length, forming short-horizon and long-horizon categories; and (3) for a given model type, the optimal TTS performance scales monotonically with compute budget. Based on these insights, we provide a practical recipe for selecting the best TTS strategy, considering problem difficulty, model type, and compute budget, providing a practical guide to effective inference-time scaling.¹

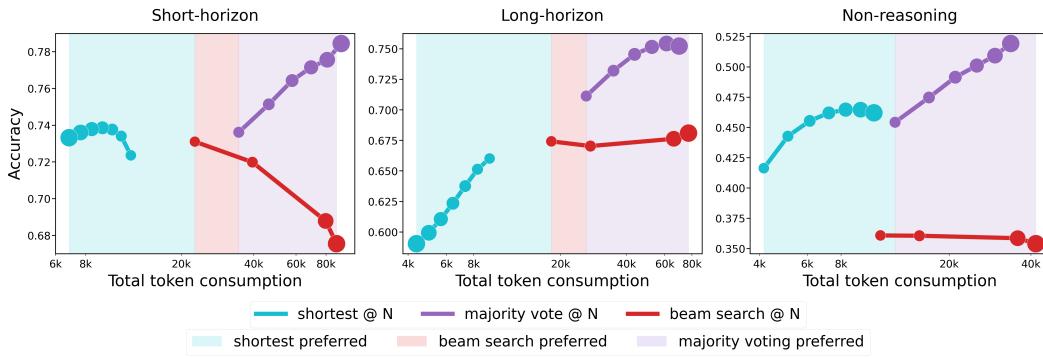


Figure 1: Plots of shortest (cyan), majority-voted (purple), and beam-searched (red) trace performances for short-horizon (left), long-horizon (middle), and non-reasoning (right) models. Short-horizon models include *R1*, *DAPO-32B*, and *QwQ-32B*; long-horizon models include *Qwen3-32B*, *GPT-OSS-120B* and *R1-32B*; and non-reasoning models include *Qwen3-235B-Instruct* and *DeepSeek-Chat*. Performance is measured using average accuracy on the AIME 2024–2025 and GPQA Diamond datasets. Shaded regions show the optimal TTS strategy by compute budget: shortest for low compute, beam search for medium, majority voting for high. The plot illustrates that there is no free lunch for TTS strategies: no single strategy is optimal and optimality depends on compute budget. This highlights the need for a principled, model-aware approach to determine the best scaling strategy at test-time. Marker size increases with N ($N \geq 2$); N is the number of parallel traces sampled.

*Corresponding author: aradhye.agarwal@gmail.com

¹The source code is available at https://github.com/Aradhye2002/art_of_tts

1 Introduction

Test-time scaling (TTS) has emerged as an effective method to enhance the reasoning capabilities of large language models (LLMs) by increasing inference-time compute. Any sound TTS strategy, by definition, will exhibit performance improvements as more compute is allocated. The best TTS strategy to choose, however, remains an open question.

Early studies explored sequential scaling methods, either by artificially extending reasoning traces (Muennighoff et al., 2025) or by encouraging deeper exploration within a single reasoning direction before switching, thereby mitigating “underthinking” (Wang et al., 2025b). More recent analyses have questioned the benefits of such sequential extensions. Notably, Gema et al. (2025) examined synthetic tasks designed to isolate specific reasoning abilities, such as counting with distractors, and regression with spurious correlations. Their findings indicate that longer reasoning can reinforce incorrect behaviors, amplify errors, and misalign reasoning paths, thereby degrading accuracy and even introducing safety concerns.

Similarly, Hassid et al. (2025) proposed *short-m@k*, a parallel TTS technique where the final prediction is obtained by majority voting among the m shortest reasoning traces, out of k sampled outputs. Their results support the idea that shorter, more concise reasoning often outperforms extended deliberation.

Prior studies, while offering valuable insights, do not account for model variations and rely on older reasoning models. In this work, we revisit these findings using more recent models, including the GPT-OSS (OpenAI et al., 2025) and Qwen3 (Yang et al., 2025) series. Our findings reveal that the relationship between compute and performance varies across model families—a divergence we attribute to differences in their post-training algorithms.

We argue that distinct post-training methods give rise to varying **reasoning horizons**. Models with large horizons (large-horizon models) are able to sustain deeper reasoning by means of longer traces, thereby benefiting in performance on harder tasks where greater thought is necessary. Short-horizon models, however, cannot generate long coherent traces, thereby making it most suitable for them to prioritize concise reasoning, irrespective of problem difficulty.

As a consequence of their training dynamics, short-horizon models commonly emerge from post-training with GRPO or GRPO-like algorithms, aligning with the well-documented length bias introduced by GRPO (Yu et al., 2025). In contrast, long-horizon models are typically produced by alternative reinforcement-learning methods that maintain stability over extended traces. For example, Qwen3—a long-horizon model—is post-trained using GSPO rather than GRPO (Zheng et al., 2025). This observation supports our hypothesis that the choice of post-training strategy plays a key role in determining a reasoning model’s effective horizon.

Overall, our work highlights the need for a model-aware perspective on TTS that accounts for differences in training methodology, problem difficulty, and compute availability to guide principled strategy selection.

2 Preliminaries

2.1 Test-time scaling methods

Test-time scaling strategies for LLMs vary widely, typically falling into parallel, sequential, hybrid/meta, and internal compute mechanisms (Figure 2). While each class of methods shows promise in specific settings, no single strategy is universally optimal.

Parallel scaling strategies improve performance by aggregating answers across multiple independently sampled reasoning traces. Self-consistency (Wang et al., 2023) samples diverse reasoning paths and chooses the most frequent final answer, significantly improving performance on arithmetic and symbolic tasks. Best-of- n sampling is widely used as a simple parallel method (Snell et al., 2024), though more principled voting strategies like majority voting (Lightman et al., 2023), and Multi-Agent Verification (MAV) (Lifshitz et al., 2025) have been recently proposed. Short-m@k (Hassid et al., 2025) exploits early stop-

ping; it runs k reasoning chains in parallel and halts early based on the proportion of traces completed.

Sequential scaling strategies extend reasoning depth by iteratively refining, restarting, or backtracking. Chain-of-Thought (CoT) prompting (Wei et al., 2023) is a fundamental idea, and subsequent work like STaR (Zelikman et al., 2022) and Reflexion (Shinn et al., 2023) explore revision through trial-and-error or verbal self-reflection. Tree-of-Thought (ToT) (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024) scale this further via structured breadth-first or DAG-style search. AlphaGeometry (Chervonyi et al., 2025) integrates symbolic proof search with LLMs for step-level sequential control. S1 (Muennighoff et al., 2025) fine-tunes models for teaching self-correction strategies, utilizing higher test-time compute.

More recent efforts like **hybrid scaling strategies** blend both axes. Meta-Reasoner (Sui et al., 2025) uses contextual bandits to dynamically select TTS strategies based on perceived task difficulty. AgentTTS (Wang et al., 2025a) and START (Li et al., 2025) deploy agents (LLMs with tool-calling capabilities) to switch between direct generation or more elaborate reasoning. PEARL (Liu et al., 2025) interleaves draft generation with refinement, simulating self-improvement loops. These meta-schedulers recognize that neither deep nor parallel scaling alone is enough, and aim to adapt the strategy based on model behavior and prompt dynamics. Internal scaling strategies, in contrast, modify how much computation the model performs internally during inference, without explicitly adjusting the number of external samples or reasoning steps. HALT-CoT (Laaouach, 2025) and Soft-CoT++ (Xu et al., 2025) estimate answer uncertainty and terminate early if confidence is high.

No strategy is universally best. Multiple empirical studies reinforce that no TTS strategy consistently dominates. Zhang et al. (2025) emphasize tradeoffs across accuracy, consistency, and efficiency—the “TTS trilemma.” Snell et al. (2024) show that compute-optimal allocation (e.g., short inference on easy questions, deeper inference on hard ones) outperforms scaling model size alone. Ghosal et al. (2025) and Hassid et al. (2025) show that longer CoT chains often degrade accuracy. Inverse-scaling effects (Gema et al., 2025) demonstrate that larger models or longer prompts may hurt, especially when uncertainty is high or symbolic reasoning is required. This underscores our central thesis: optimal TTS is highly contextual and must consider model training (e.g., type of post-training), task type, and difficulty.

In this work, we consider first finish search (FFS, Algorithm 1), last finish search (LFS, Algorithm 2) and beam search for our analyses, the first two of which are parametrized by variables k and N , while the last is parametrized by N alone. FFS- $k@N$ means sampling N outputs and performing MV among the shortest k samples to determine the majority vote while LFS- $k@N$ simply involves choosing the longest k samples instead of shortest, followed by majority voting on these. Beam search involves maintaining a beam of high probability partial hypotheses, continuously updating these prefixes as decoding progresses.²

2.2 Models

We evaluate both reasoning and non-reasoning models to analyze the effects of TTS strategies across diverse training paradigms.

Reasoning models. DeepSeek-R1 is a reasoning-tuned LLM optimized for mathematical and logical tasks using GRPO—an RL algorithm that improves efficiency over PPO

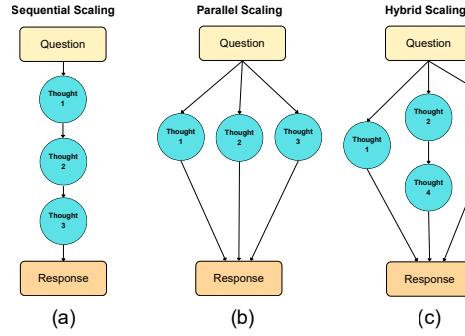


Figure 2: Different TTS paradigms

²Since we use API-based model inference (deepinfra.com), we restrict our analysis to API-friendly TTS strategies.

Algorithm 1 First Finish Search - k (FFS-k)

Require: Model M , prompt x , number of samples N , filter size k
Ensure: Final answer y^*

- 1: Generate N outputs $\{y_1, \dots, y_N\}$ in parallel
- 2: Stop as soon as k traces are complete
- 3: Select these k traces $\{y_{(1)}, \dots, y_{(k)}\}$
- 4: Extract final answers from these k traces
- 5: **return** majority-voted answer among them

Algorithm 2 Last Finish Search - k (LFS-k)

Require: Model M , prompt x , number of samples N , filter size k
Ensure: Final answer y^*

- 1: Generate N outputs $\{y_1, \dots, y_N\}$ in parallel
- 2: Sort **completed** outputs by trace length (descending)
- 3: Select longest k traces $\{y_{(1)}, \dots, y_{(k)}\}$
- 4: Extract final answers from these k traces
- 5: **return** majority-voted answer among them

but introduces biases in gradient normalization, leading to uneven penalization across trace lengths. **R1-32B** is a distilled 32B-parameter variant of DeepSeek-R1 that inherits its reasoning-centric behavior, exhibiting similar trace-length-dependent trends at reduced capacity. **QwQ-32B** is a reasoning-focused model from Qwen that leverages stronger MoE routing, typically producing shorter and more compact reasoning traces. **GPT-OSS-120B** is a large open-source GPT-style model trained with extensive reasoning supervision, serving as a transparent large-scale baseline. **Qwen3-32B** belongs to the Qwen3 family and emphasizes diverse reasoning domains—STEM, code, and commonsense—yielding qualitatively distinct reasoning patterns from DeepSeek models. **DAPO-32B** is a RL-trained reasoning model based on the DAPO algorithm, an open-source alternative to GRPO that claims to mitigate its gradient normalization bias while maintaining sample efficiency.

Non-reasoning models. **Qwen3-235B-Instruct** is a large instruction-tuned MoE model (235B total, 22B active parameters) without explicit reasoning supervision, producing fluent but unstructured responses. **DeepSeek-Chat** is the general-purpose conversational model from DeepSeek, optimized for dialogue and summarization rather than multi-step reasoning, allowing us to assess the impact of TTS on models without reasoning-centric training.

2.3 Datasets

We evaluate models on two complementary reasoning benchmarks—**AIME** and **GPQA Diamond**—which together cover both symbolic-numerical and conceptual reasoning domains.

The **American Invitational Mathematics Examination (AIME)** is a high-school level contest assessing symbolic and arithmetic reasoning through 30 short-answer problems, each with an integer solution between 0 and 999. We use three recent variants—**AIME 2024**, **AIME 2025-I**, and **AIME 2025-II**—to test consistency across different years and question distributions. Each problem is formatted as a concise natural-language prompt, and models are instructed to output the final answer within “\boxed{}” for consistent evaluation. AIME problems typically require multi-step deductive reasoning, involving algebraic or combinatorial manipulation, making them ideal for analyzing the accuracy-efficiency trade-offs of TTS strategies.

GPQA Diamond (Rein et al., 2023) is a graduate-level benchmark designed to test conceptual and factual reasoning across physics, biology, and chemistry. Each question is multiple-choice with four options (A–D), and models must output the selected answer in a “\boxed{}” format for standardized parsing. We employ the **Diamond** subset, the most challenging and expert-verified split, emphasizing high conceptual depth and factual precision. In contrast to AIME’s numerical reasoning focus, GPQA Diamond evaluates abstract and knowledge-grounded reasoning. Together, they provide a comprehensive view of reasoning performance across mathematical, symbolic, and conceptual domains.

All model- and dataset-specific hyperparameters are listed in Appendix A.

2.4 Metrics

Accuracy. This metric is defined as the proportion of generated traces whose final prediction matches the ground-truth answer. Even when parsing is reliable, and there is only a

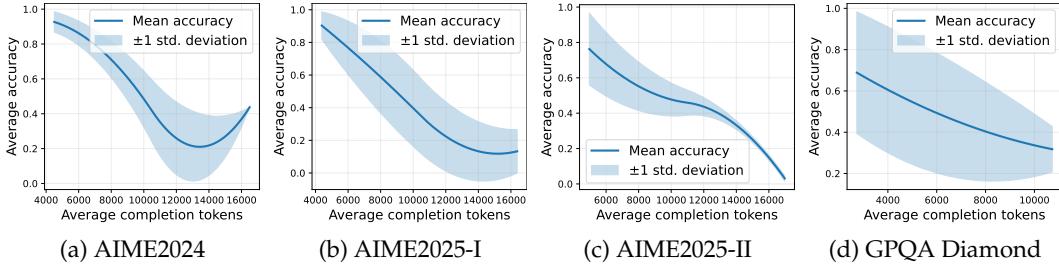


Figure 3: Mean accuracy vs. average completion tokens for different datasets averaged across all models

single gold answer, representational ambiguities can arise. For instance, the fraction $\frac{1}{2}$ may be written as “1/2” or “0.5,” both of which are semantically equivalent but syntactically distinct. In our setting, accuracy evaluation is simplified because three of the four datasets are derived from AIME, where answers are restricted to three-digit integers. The remaining dataset (GPQA) is multiple-choice with options limited to A–D. For both cases, we explicitly instruct the model to provide its final answer within delimiters, which facilitates reliable extraction of the predicted value.

Token consumption. We count token consumption in two distinct ways, capturing different aspects of the utilized compute. **Total tokens** refer to the total number of tokens generated across all the traces in order to arrive at the generated answer. **Sequential tokens**, on the other hand, refer to the number of tokens that must necessarily be produced in a sequence, and are dependent on the previously generated tokens. For instance, during vanilla decoding using greedy or stochastic sampling, generating a trace x would involve a sequential token count of $|x|$ since x_i can only be generated after all of $X_j, j < i$ are generated. For an inference strategy which requires generating N complete traces x^1, x^2, \dots, x^N , the sequential token count would be $\max_{i=1}^N |x^i|$ since all the tokens in the longest trace would be dependent on the one before it. Therefore, while total tokens measure the overall compute used, sequential token count gives an estimate of the minimum possible latency in generating a given output (assuming each token generation takes the same time).

2.5 Measuring Problem Difficulty

In order to devise a granular recipe for the appropriate scaling strategy to use at test-time, it is crucial to take into account the problem difficulty. The direct way to measure difficulty of a question is to simply calculate the task accuracy for a given problem, averaged across all models and sampled traces. Another, more indirect approach would be to calculate the average tokens generated for the task, again averaged across all models and outputs. Interestingly, we find that both these metrics are correlated (Figure 3), and that this overall trend holds across all datasets, where a reasoning (as well as a non-reasoning) models “thinks” longer on harder (as measured through accuracy) problems.

Finding

Both reasoning and non-reasoning models think longer for harder problems.

While both reasoning and non-reasoning models expend more tokens on harder problems, longer trace lengths alone do not guarantee improved quality. Recent research suggests that excessive deliberation can harm performance by propagating early mistakes, contributing to the growing view that more compute is not always better (Gema et al., 2025).

3 Results

3.1 Beam search shows inverse or no scaling

We notice that across two of the model families—short-horizon and non-reasoning—beam search exhibits a consistent inverse-scaling pattern: *performance degrades monotonically as*

Table 1: Model categorization, behavioral characteristics, and accuracy as a function of trace length and problem difficulty. Tasks are classified as easy or hard based on whether their difficulty is below or above the median across all tasks. Trace lengths are labeled short or long using the model-specific median trace length computed over the entire task set.

Category	Model	Behavior	Accuracy			
			Easy		Hard	
			Short	Long	Short	Long
Short horizon	R1	Shorter is always better	0.95	0.72	0.61	0.48
	DAPO-32B		0.80	0.54	0.05	0.05
	QwQ-32B		0.91	0.70	0.58	0.58
Long horizon	GPT-OSS-120B	Shorter is better for easy problems while longer is better for hard problems	0.92	0.85	0.48	0.53
	Qwen3-32B		0.75	0.63	0.22	0.45
	R1-32B		0.92	0.62	0.33	0.34
Non-reasoning	Qwen3-235B	Shorter is always better	0.90	0.52	0.51	0.20
	DeepSeek		0.47	0.22	0.12	0.06

the beam size N increases (Figure 1). For short-horizon models such as R1 and QwQ-32B, accuracy drops sharply once N becomes larger than 2; for non-reasoning models there is a similar, although milder, trend of performance drops as N increases. Even long-horizon models like GPT-OSS-120B, and Qwen3-32B fail to benefit from beam expansion: their accuracy curves flatten or decline as N increases. Since total token consumption—and therefore total compute—increases with beam width, these results reveal a clear case of *inverse compute scaling*, where allocating more test-time compute via larger beams either harms accuracy or yields no benefit.

Finding

Beam search performance degrades or remains the same with increasing beam size for reasoning-focused datasets like AIME and GPQA.

3.2 Correlation of trace length with quality

It is crucial to understand how the trace length correlates with quality (as measured through accuracy) in order to obtain a deeper understanding of length-based filtering strategies like FFS and LFS. FFS and LFS are based on two diametrically opposite viewpoints: *shorter is better* and *longer is better*. To investigate which hypothesis (or hypotheses) hold for a given model, we report the accuracy for a given interval of trace lengths and problem difficulties (Table 1). Note that the problem difficulty is measured by averaging the accuracy over all models and traces (Section 2.5), while the reported accuracy is measured by averaging over all outputs for the specific model. A key consideration is that problem difficulty is confounded with trace length (Figure 3): short traces typically arise from easier problems, whereas long traces tend to correspond to harder ones. To mitigate this confounding effect, we restrict our analysis to tasks for which both short and long traces are available. For each such dataset, we compute a single accuracy value for short and long traces separately, and then average these values across datasets, thereby preventing differences in dataset size from disproportionately influencing the aggregated results. Based on the ordering between these reported accuracies, we broadly classify the six reasoning models as either short-horizon or long-horizon. While the two non-reasoning models both show short-horizon behavior, we choose to keep them separate from short-horizon models due to the significant differences in the post-training techniques employed (instruction tuning vs. RL).

Across all models, we observe a consistent invariant: for any given trace-length bucket, the reported accuracy is always higher on easy problems than on hard ones. This pattern is expected, as problem difficulty is defined through aggregated accuracy, and harder questions naturally exhibit lower correctness rates.

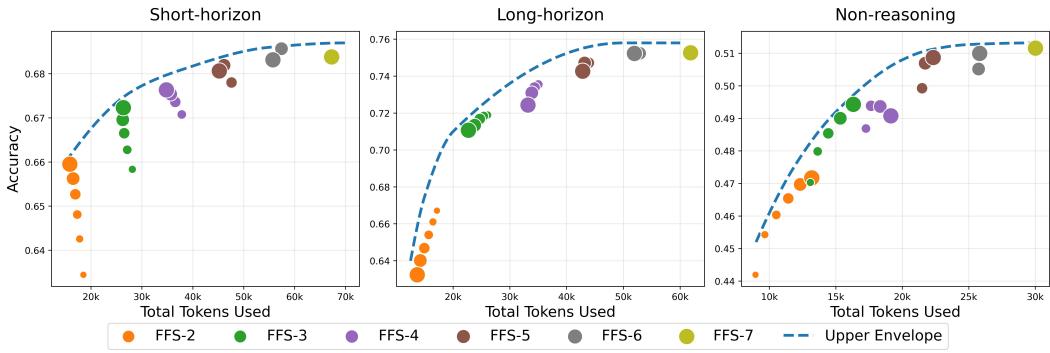


Figure 4: Accuracy versus token usage for different model families. FFS- k variants are shown in distinct colors (one color per k). Marker size encodes the value of N , with larger markers representing larger N .

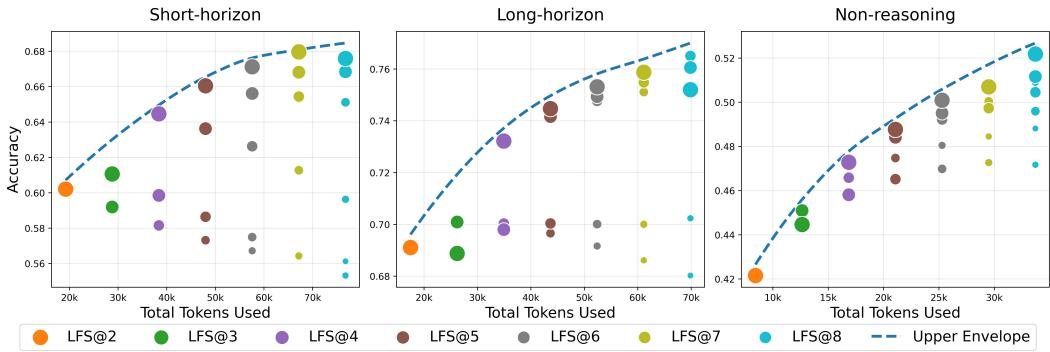


Figure 5: Accuracy versus token usage for different model families. LFS@ N variants are shown in distinct colors (one color per N). Marker size encodes the value of k , with larger markers representing larger k .

It is more interesting to observe how the order between short and long traces for easy and hard problems varies across different models. For short-horizon models (R1, QwQ-32B, DAPO-32B), we find that for a given problem difficulty, shorter traces are more likely to be correct than longer ones. This is in line with the recent observations made by (Agarwal et al., 2025; Hassid et al., 2025) where the authors find that conciseness in the reasoning trace is linked to better accuracy. However, we observe a different phenomenon with other more advanced models such as Qwen3-32B and GPT-OSS-120B, where for easier problems shorter traces are better, but for harder problems longer traces are preferred.

DAPO-32B shows a similar length bias pattern to prior models, with shorter traces more likely to be correct than longer ones (Table 1). The bias level is also close to that of R1, which suggests that any improvements in mitigating length bias over GRPO may be limited under our evaluation.

The complete results for different models, datasets, and TTS strategies can be found in Appendix B. Individual, model-wise plots for FFS and LFS are present in Appendix C.

Finding

DAPO induces length bias to the same extent as GRPO.

4 Analysis

It is necessary to determine how the performance of FFS- k @ N and LFS- k @ N varies for different values of k and N across the models, in order to find the optimal strategy. Figures 4

Model Family	Difficulty	Compute	Recommended Recipe
Short-horizon	High / Low	High	MV@N; N large
	High / Low	Low	FFS-k@N; k=1, N large
Long-horizon	High / Low	High	MV@N; N large
	High / Low	Low	SD
Non-reasoning	High / Low	High	MV@N; N large
	High / Low	Low	FFS-k@N; k=1, N large

Table 2: Decision matrix outlining optimal TTS strategies based on model family, task difficulty, and computational budget. K denotes the number of shortest/longest traces considered for voting, and N indicates the total trace count. SD refers to simple decoding, a greedy left-to-right generation procedure analogous to beam search with beam size 1: at each generation step, the model selects only the single most probable continuation.

and 5 depict the performance of FFS-k@N and LFS-k@N for the different model types. These plots reveal an interesting behavior where the optimal TTS strategy always seems to scale with increasing budget.

Furthermore, we find that for the LFS family of methods, the maximum performance for a given amount of total compute is always achieved when k is large (which implies k=N). Note that k=N is simply MV-N, and therefore we conclude that MV@N is better than LFS-k@N for any value of k, all while consuming the same number of tokens.

Finding

LFS is always suboptimal to MV: longest-trace filtering consistently reduces accuracy at the same compute.

For the FFS family of methods, we observe a more nuanced behavior where while performance improves for increasing k (while also consuming more tokens) across all model types, the behavior with N for a fixed k is mixed. We find that for short-horizon models larger values of N are always best (higher performance at lesser token consumption), while for long-horizon and non-reasoning models there is a tradeoff between performance and the compute consumed. Note that while a tradeoff exists for both long-horizon and non-reasoning models, the handles to vary the tradeoff are opposite for them: for long-horizon models to draw performance at the cost of higher compute one has to choose smaller N (essentially performing simple decoding) while for non-reasoning models one has to choose larger N.

5 The Recipe

Our analysis reveals that the optimal test-time scaling strategy is not universal but depends on a combination of the model’s architectural family, the difficulty of the problem at hand, and the available compute budget. To distill our findings into actionable guidance, we present a decision matrix in Table 2. We explain below the rationale for choosing such a recipe below.

Short-horizon models. Across both low- and high-difficulty settings, short-horizon models consistently prefer shorter traces over longer ones (Table 1). Because FFS-k improves in both accuracy and computational cost as k increases, we select small values of k (specifically k = 1) under low-compute constraints, and large values of k (namely k = N) when ample compute is available. The latter choice is equivalent to MV@N, since selecting the k = N shortest traces from N samples necessarily involves including all traces. Additionally, for short-horizon models, performance increases with larger N for any fixed k. Accordingly, we choose N to be as large as permitted by the compute budget.

Long-horizon models. For high-difficulty settings, long-horizon models prefer longer traces. Because LFS@N improves as N increases, we use large N when compute is abundant and small N (ideally N = 1) when compute is limited. Keeping N fixed, performance

increases with larger k ; thus, in both compute regimes we set $k = N$, which corresponds to MV@N. Under low-compute conditions where $N = 1$, MV@N reduces to simple decoding (SD) without any aggregation.

For low-difficulty settings, we instead use FFS, since long-horizon models prefer shorter traces for easier problems. As with short-horizon models, FFS- k scales positively with k , so we employ a large k when compute is high and a small k when compute is low. In these settings, performance improves as N decreases (in contrast to short-horizon models, where larger N is beneficial). Therefore, we set $N = k$, which yields the MV@N strategy. Under low compute, this results in $k = 1$ and thus simple decoding (SD), while under high compute it corresponds to MV with a large sample size.

Interestingly, although the model types exhibit distinct behavior across different task difficulties, the optimal TTS strategy is ultimately independent of the problem difficulty, as shown in the final recipe (Table 2).

Finding

The optimal TTS strategy is independent of task difficulty.

6 Conclusion

Our large-scale study demonstrates there is no single optimal test-time scaling (TTS) strategy for enhancing LLM reasoning. The most effective approach is contingent on a crucial interplay between the model’s training methodology, problem difficulty, and the available compute budget. We find that different model families exhibit distinct behaviors: **short-horizon** models consistently favor shorter, concise traces, while **long-horizon** models benefit from longer, more deliberate reasoning for harder problems while concise reasoning for easier problems. Critically, beam search consistently proves suboptimal for complex reasoning. Our work provides a practical framework for practitioners, underscoring that maximizing performance requires a nuanced, model-aware approach to inference rather than a universal strategy.

References

- Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. First finish search: Efficient test-time scaling in large language models, 2025. URL <https://arxiv.org/abs/2505.18149>.
- Markus Besta, Nora Blach, Alexander Kubicek, Robin Gerstenberger, Malte Podstawska, et al. Graph of thoughts: Solving elaborate problems with large language models. In AAAI 2024, 2024.
- Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V. Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2, 2025. URL <https://arxiv.org/abs/2502.03544>.
- Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Ardit, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. Inverse scaling in test-time compute, 2025. URL <https://arxiv.org/abs/2507.14417>.
- Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? mirage of test-time scaling in reasoning models, 2025. URL <https://arxiv.org/abs/2506.04210>.
- Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don’t overthink it: preferring shorter thinking chains for improved llm reasoning, 2025. URL <https://arxiv.org/abs/2505.17813>.

Yassir Laaouach. HALT-cot: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy. In *4th Muslims in ML Workshop co-located with ICML 2025*, 2025. URL <https://openreview.net/forum?id=CN5c7C1CZa>.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. Start: Self-taught reasoner with tools. *arXiv preprint arXiv:2503.04625*, 2025.

Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers. *arXiv preprint arXiv:2502.20379*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. Pearl: Parallel speculative decoding with adaptive draft length, 2025. URL <https://arxiv.org/abs/2408.11850>.

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, 2025. URL <https://arxiv.org/abs/2501.19393>.

OpenAI, ;, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sébastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL <https://arxiv.org/abs/2508.10925>.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpq: A graduate-level google-proof q&a benchmark, 2023. URL <https://arxiv.org/abs/2311.12022>.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yuan Sui, Yufei He, Tri Cao, Simeng Han, Yulin Chen, and Bryan Hooi. Meta-reasoner: Dynamic guidance for optimized inference-time reasoning in large language models. *arXiv preprint arXiv:2502.19918*, 2025.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, and Suhang Wang. Agenttts: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. *arXiv preprint arXiv:2508.00890*, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL <https://arxiv.org/abs/2203.11171>.

Yue Wang, Qiuwei Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Thoughts are all over the place: On the underthinking of o1-like llms, 2025b. URL <https://arxiv.org/abs/2501.18585>.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL <https://arxiv.org/abs/2201.11903>.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning, 2025. URL <https://arxiv.org/abs/2505.11484>.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yingren Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, and Thomas Lee. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS 2023*, 2023.

Qiyi Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL <https://arxiv.org/abs/2503.14476>.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

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>.

A Hyperparameters

All hyperparameters for our experiments are given in Table 3.

Model	GPQA	AIME24	AIME25-I	AIME25-II	Top- p	Temp.
Deepseek	16K	32K	32K	32K	0.95	0.6
R1	32K	32K	32K	32K	0.95	0.6
QwQ	32K	32K	32K	32K	0.95	0.6
R1-Distill-Qwen	32K	32K	32K	32K	0.95	0.6
GPT-OSS-120B	8K	8K	8K	8K	0.95	0.6
Qwen3-235B	5K	5K	5K	5K	0.95	0.6
Qwen3	16K	32K	32K	32K	0.95	0.6
Dapo-Qwen-32B	10.1K	20.5K	20.5K	20.5K	0.7	1.0
<i>Global settings (shared across all models)</i>						
Beam width				8		
Samples (n , MV/LFS/FFS)				8		
Answer-reserve for BF				3K		

Table 3: Decoding hyperparameters used in all experiments across models and datasets. Identical values across datasets are shown once.

B Results

Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS
Seq. tokens	–	9.2k	9.2k	3.0k	Seq. tokens	6.3k	13.3k	13.3k	0.8k	Seq. tokens	4.4k	5.0k	5.0k	3.0k
Total tokens	–	45.4k	45.4k	4.3k	Total tokens	50.3k	31.6k	31.6k	0.9k	Total tokens	34.8k	32.0k	32.0k	5.9k
GPQA	–	54.0	53.5	55.1	GPQA	56.6	58.1	53.0	48.5	GPQA	71.7	72.2	66.2	66.2
AIME24	–	53.3	46.7	53.3	AIME24	26.7	43.3	36.7	26.7	AIME24	73.3	80.0	80.0	76.7
AIME25-I	–	46.7	40.0	33.3	AIME25-I	26.7	46.7	33.3	33.3	AIME25-I	60.0	80.0	73.3	66.7
AIME25-II	–	46.7	33.3	40.0	AIME25-II	20.0	20.0	26.7	13.3	AIME25-II	80.0	86.7	86.7	86.7
(a) Dapo-Qwen-32B					(b) Deepseek-Chat					(c) GPT-OSS-120B				
Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS
Seq. tokens	13.4k	18.3k	18.3k	9.6k	Seq. tokens	12.4k	19.3k	19.3k	3.5k	Seq. tokens	4.1k	5.7k	5.7k	3.8k
Total tokens	107k	106k	106k	10.4k	Total tokens	98.9k	89.6k	89.6k	3.9k	Total tokens	33.1k	35.8k	35.8k	16.7k
GPQA	68.2	66.7	61.6	66.2	GPQA	69.2	69.2	66.7	63.1	GPQA	66.7	70.7	64.1	71.2
AIME24	76.7	83.3	76.7	80.0	AIME24	83.3	90.0	83.3	40.0	AIME24	33.3	83.3	83.3	83.3
AIME25-I	60.0	73.3	53.3	60.0	AIME25-I	66.7	73.3	73.3	40.0	AIME25-I	40.0	53.3	53.3	53.3
AIME25-II	60.0	86.7	66.7	80.0	AIME25-II	80.0	86.7	73.3	40.0	AIME25-II	13.3	40.0	40.0	40.0
(d) QwQ-32B					(e) Qwen3-32B					(f) Qwen3-235B				
Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS	Metric	BS	MV	LFS	FFS
Seq. tokens	8.7k	14.6k	14.6k	6.4k	Seq. tokens	12.2k	17.9k	17.9k	6.0k	Seq. tokens	19.7k	87.9k	87.9k	6.6k
Total tokens	69.8k	78.5k	78.5k	6.7k	Total tokens	59.6	64.6	60.1	52.5	Total tokens	59.6	64.6	60.1	52.5
GPQA	72.2	74.2	71.7	73.7	GPQA	70.0	83.3	70.0	86.7	GPQA	60.0	80.0	53.3	83.3
AIME24	70.0	83.3	70.0	86.7	AIME24	46.7	60.0	46.7	53.3	AIME24	46.7	80.0	46.7	53.3
AIME25-I	60.0	73.3	53.3	66.7	AIME25-I	66.7	86.7	46.7	53.3	AIME25-I	66.7	60.0	53.3	40.0
AIME25-II	73.3	86.7	46.7	73.3	AIME25-II	80.0	60.0	53.3	40.0	AIME25-II	80.0	60.0	53.3	40.0
(g) R1					(h) R1-Distill-Qwen									

Table 4: Accuracy (%) and compute cost ($\times 10^3$ tokens) across models. For each method, token counts are dataset-averaged. Bold, gray cells mark the best value per row. Methods shown are beam search (BS), majority voting (MV), first finish search (FFS), last finish search (LFS).

Overall performance patterns. The four decoding strategies exhibit complementary trade-offs. MV is the most consistent accuracy-oriented method but incurs the largest token costs (often an order of magnitude higher than FFS). BS occasionally attains top accuracy

on specific rows but is typically expensive. LFS shows mixed reliability: it matches or surpasses MV on some symbolic tasks (e.g., AIME variants for GPT-OSS-120B and Qwen3-235B) but underperforms on several models. FFS delivers substantial token savings (commonly reducing MV token usage by tens of percent up to $\sim 90\%$), yet its accuracy impact is model-dependent—competitive in some cases (e.g., Dapo-Qwen-32B, Qwen3-235B, certain R1 entries) and substantially lower in others (e.g., DeepSeek, Qwen3).

Model-family comparison

- **Qwen-derived models (Qwen3-32B, Qwen3-235B-Instruct):** MV and LFS often lead on math tasks (AIME24/25), while FFS gives the largest efficiency gains but is not uniformly the accuracy winner.
- **DeepSeek family (DeepSeek, R1, R1-Distill-Qwen):** MV is the most stable high-accuracy choice; LFS sometimes rivals MV, and FFS reduces compute dramatically but with variable accuracy trade-offs.
- **General-purpose large models (GPT-OSS-120B, QwQ-32B):** MV/LFS usually secure top accuracy; FFS offers strong efficiency with mixed accuracy effects (near-match in some cases, notable drops in others).

Task-level behavior. GPQA is relatively robust across decoding methods (small accuracy spread); FFS often preserves GPQA performance while saving tokens. AIME24 and the AIME25 subsets are more sensitive: MV and LFS usually perform better on structured symbolic reasoning, whereas FFS can be competitive for certain large models but may degrade accuracy for others.

Takeaway. There is no single best decoding method across models and tasks. MV remains the safest option for accuracy at higher compute cost; LFS is a viable middle ground for symbolic problems; BS is occasionally useful but expensive; FFS is attractive when compute is constrained but requires careful evaluation per model/task due to mixed accuracy outcomes.

C Model-wise Plots

Figures 6 and 7 contain the FFS and LFS curves, for each of the eight models individually.

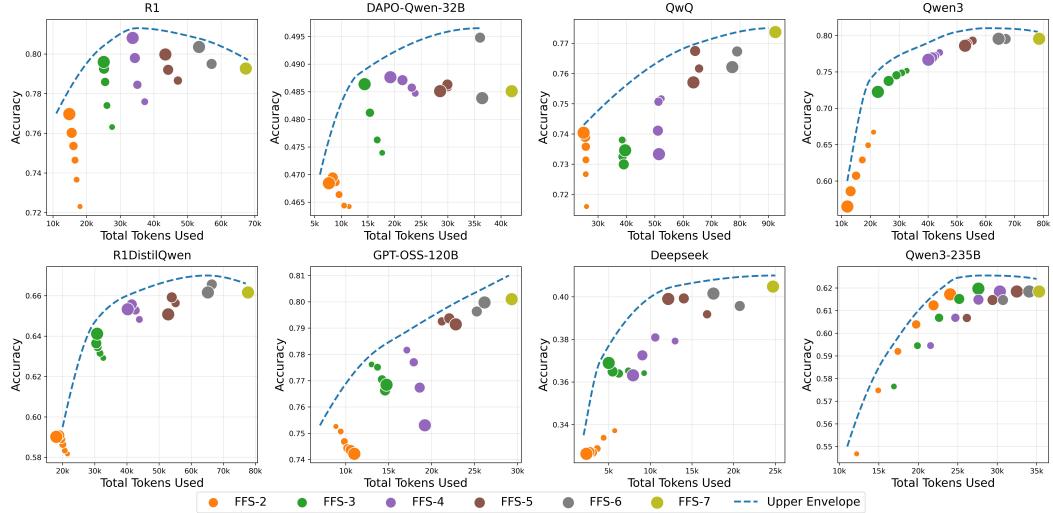


Figure 6: Accuracy versus token usage for different models. FFS- k variants are shown in distinct colors (one color per k). Marker size encodes the value of N , with larger markers representing larger N .

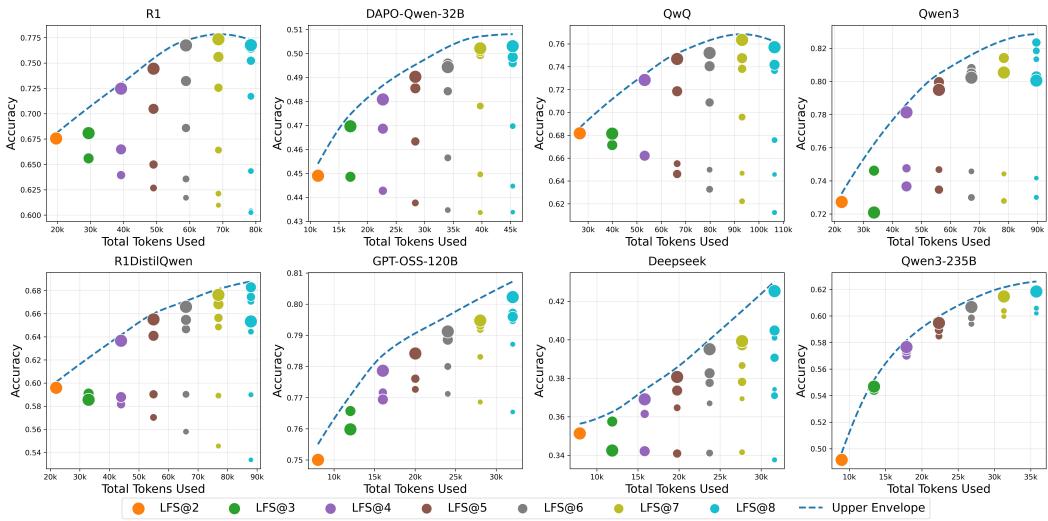


Figure 7: Accuracy versus token usage for different models. LFS@N variants are shown in distinct colors (one color per N). Marker size encodes the value of k, with larger markers representing larger k.