Where do Reasoning Models make a Difference? Follow the Reasoning Leader for Efficient Decoding

Ming Li[1,2], Zhengyuan Yang[2], Xiyao Wang[1], Dianqi Li, Linjie Li[2], Kevin Lin[2], Tianyi Zhou[1] , Lijuan Wang[2]
[1]University of Maryland [2]Microsoft
minglii@umd.edu

Abstract

Large reasoning models (LRMs) achieve strong reasoning performance by emitting long chains of thought. Yet, these verbose traces slow down inference and often drift into unnecessary detail, known as the overthinking phenomenon. To better understand LRMs' behavior, we systematically analyze the token-level misalignment between reasoning and non-reasoning models. While it is expected that their primary difference lies in the stylistic "thinking cues", LRMs uniquely exhibit two pivotal, previously under-explored phenomena: a ***Global Misalignment Rebound***, where their divergence from non-reasoning models persists or even grows as response length increases, and more critically, a ***Local Misalignment Diminish***, where the misalignment concentrates at the "thinking cues" each sentence starts with but rapidly declines in the remaining of the sentence. Motivated by the *Local Misalignment Diminish*, we propose ***FoReaL-Decoding***, a collaborative fast-slow thinking decoding method for cost-quality trade-off. In FoReaL-Decoding, a Leading model leads the first few tokens for each sentence, and then a weaker draft model completes the following tokens to the end of each sentence.

1 Introduction

Reasoning has become a pivotal capability of large language models (LLMs), driving rapid progress in mathematical problem solving, code generation, and commonsense question answering [22, 33, 2, 64, 65]. Contemporary Large Reasoning Models (LRMs) such as OpenAI's GPT-o1 [47] and the open-source DeepSeek-R1 [16] demonstrate this trend by producing explicit long chains of thought (CoT) [66] that markedly improve performance on challenging tasks in mathematics [70, 69], programming [41], and other complex domains. These deeper, longer, and more precise reasoning trajectories are cultivated by reinforcement-learning-based optimization [16] or supervised fine-tuning on expert demonstrations [74, 46, 32], representing advanced "slow-thinking" patterns [24, 33, 37]. Although these slow-thinking LRMs showcase impressive reasoning skills, communities are increasingly concerned about the efficiency and fidelity of their often-lengthy chains of thought, a phenomenon known as "overthinking" [12, 18], where excessive computational resources are allocated for simple problems with minimal benefit.

To alleviate overthinking and improve efficiency, a series of methods has been proposed [75, 57, 1, 68, 45, 21, 55, 54, 76, 19, 71, 51, 56, 61, 67]. Most of these, however, require further post-training or manipulate the LRM's distribution itself, adding complexity or computational overhead. Motivated by Speculative Decoding [26] and the distinctions between fast and slow thinking, we ask: *Is it possible to design a collaborative, training-free decoding method that mixes fast and slow thinking models to effectively trade-off quality and efficiency?*

To answer this and develop such a method, we first seek to pinpoint what truly differentiates strong reasoning models from standard instruction-following LLMs at the token level. For instruction-

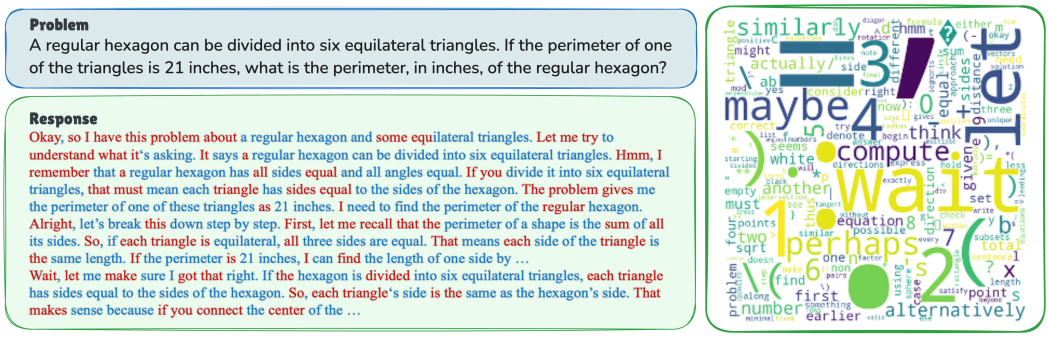

Figure 1: **Left**: An example comparing the token distribution alignment between *DeepSeek-R1-Distill-Qwen-32B* and *Qwen2.5-1.5B-Instruct*, qualitatively showing that the misaligned tokens (colored in red) are related to thinking patterns, and probably appear at the start of sentences. **Right**: The WordCloud of the misaligned tokens calculated on a mix of math datasets, quantitatively showing the high-frequency misaligned tokens like "*wait*", "*perhaps*", "*maybe*", "*let*", and "*alternatively*".

following models, LIMA [77] proposes the "superficial alignment" hypothesis, in which it shows that most of the knowledge has been learned in the pretraining and only a small amount of data is needed for alignment. Although a line of work tries to use various methods for data selection on either instruction-following [9, 35, 42, 34] or reasoning [46, 74] capabilities, [40] verifies this hypothesis from token-level analysis between the base model and the aligned model.

Leveraging the diagnostic framework of [40], our systematic analysis of misalignment across various model types (large reasoning, small reasoning, instruction-following, and pretrained base model) reveals critical insights. We observe a "superficial alignment" phenomenon similar to [40], where misaligned tokens are predominantly stylistic (e.g., "*Hmmm*", "*Wait*", "*Let me check*") rather than content-specific, often related to explicit thinking patterns. More strikingly, while previous work showed that misalignment between instruction-following and base models diminishes with longer context, we find this does not hold for reasoning models. Instead, we identify a ***Global Misalignment Rebound***, where overall misalignment between reasoning and non-reasoning models can slightly grow with response length, suggesting that increasing the length cannot reduce the misalignment. This indicates that the reasoning abilities are *not* as superficial as instruction-following. Crucially, despite this global trend, we uncover a corresponding ***Local Misalignment Diminish*** phenomenon: most token misalignments occur at the *beginning of each sentence*, then rapidly decrease until the next sentence starts. These findings reveal a novel *periodical, sentence-level misalignment diminishing pattern* unique to LRMs, driven by thinking-pattern indicators concentrated at sentence openings, shedding light on a better understanding of token-level divergences of these two types of models.

Based on this core insight that the reasoning pattern of LRMs is often front-loaded in each sentence, we hypothesize that strategic, limited intervention by a strong LRM can guide a weaker model, balancing reasoning quality with efficiency. To this end, we propose ***Follow the Reasoning Leader (FoReaL-Decoding)***, an efficient collaborative decoding method. In FoReaL-Decoding, a strong Leading model generates the initial few tokens of each sentence (capturing the potentially misaligned "thinking cues"), after which a weaker Draft model completes the sentence. To further mitigate potential overthinking from the Leading model (e.g., endlessly generating "Wait"), we introduce a stochastic binary gate that controls whether the Leading model intervenes on a given sentence. These two control knobs, lead token count and lead probability, allow FoReaL-Decoding to smoothly interpolate between the Draft and Leading models, offering strong controllability over the cost-quality spectrum.

2 Token Distributions of Reasoning vs. Non-Reasoning Models

Large-scale reasoning models (LRMs) often outperform smaller instruction-tuned models on complex reasoning-heavy tasks, yet how their generation behavior differs from instruction models within the same model family remains unclear. [40] proposes an analytical method through the lens of token-distribution shifts and finds that alignments between instruction-following and base pretrained models are often superficial. This phenomenon is supported by nearly identical decoded tokens

in the majority of token positions under the same input contexts, with distribution shifts occurring mainly with stylistic tokens like discourse markers. However, the critical question remains: "*Does this superficial alignment finding on instruction-following LLMs still hold for today's capable LRMs?*" Thus, our work systematically investigates token misalignment across various model combinations involving LRMs.

In the analysis, the aligned positions are defined as those token steps where the Draft model, when conditioned on the Leading model's history, would greedily generate exactly the same token as the Leading model, which means that *the two models have the same most probable behavior under the same context, indicating the alignment*.

Qualitative Analysis on Misaligned Tokens. Figure 1 (left) shows a qualitative example (truncated) from MATH500, comparing the token distribution alignment between *DeepSeek-R1-Distill-Qwen-32B* as the Leading model and *Qwen2.5-1.5B-Instruct* as the Draft model. The shown response y is generated by the Leading model, the aligned tokens $\mathbf{y}_{\mathcal{A}}$ are colored in blue, and misaligned tokens $\mathbf{y}_{\mathcal{A}^\complement}$ are colored in red. Through the example, it can be intuitively perceived that the misaligned tokens are mostly stylistic tokens related to thinking patterns, and the beginning of each sentence has a larger probability of being misaligned. To further quantitatively investigate what exactly these misaligned tokens are, we extract all the misaligned tokens from the mix of AIME24, AMC23, GPQA, and MATH datasets, count their frequencies, and generate the corresponding WordCloud shown in Figure 1 (right). From the WordCloud, it is observed that most of the high-frequency misaligned tokens are related to thinking patterns of LRMs, like "*wait*", "*perhaps*", "*maybe*", "*let*", and "*alternatively*", which shows a similar but different superficial phenomenon than previous instruction-following LLMs: While misalignment in both types of models is primarily stylistic rather than content-based, those in LRMs are distinctively characterized by tokens reflecting their overt reasoning or self-correction patterns. Thus, our qualitative exploration reveals that LRM misalignment is characterized by stylistic "thinking cues" concentrated at sentence beginnings, prompting a more detailed quantitative analysis of their underlying distribution patterns.

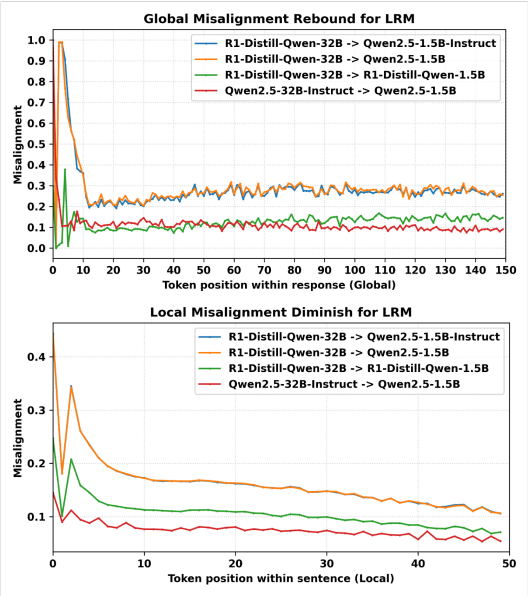

Figure 2: **Top**: Response-level misalignment changes with response length. **Bottom**: sentence-level misalignment changes with response length. The y-axis represents the average misalignment rate at each token position, the x-axis represents the token position within the whole response or sentence. We reveal the novel Global Misalignment Rebound and Local Misalignment Diminish phenomenon that only occurs on current LRMs, shown as the blue, orange, and green lines of the upper figure. This phenomenon does not hold for the previous alignment between the instruction-following and base models, shown in the red line.

Global Misalignment Rebound. Existing analysis on token distribution shifts between instruct and base models has identified that such shifts will gradually diminish over time during the decoding process due to the more comprehensive context given, as shown in Figure 2 (upper, red line). In the figure, the y-axis represents the average misalignment rate at each token position, while the x-axis represents the token position within the whole response (upper panel) or sentence (lower panel). As shown, the red line, representing misalignment between the instruct model and base model, decreases and remains at a low rate. This implies that providing longer context can gradually compensate for the misalignment between instruct and base models.

However, this response-level misalignment diminishing phenomenon does not strictly hold for LRMs. As illustrated in Figure 2 (upper), lines corresponding to LRM as the Leading model exhibit different behaviors. When the Draft models are instruct (blue line) or base (orange line) models, the misalignment rates initially decrease dramatically to around 0.2, then rebound and persist around 0.3. In contrast, the green line, representing misalignment between large and small reasoning

models (which belong to the same family and are trained on similar data), shows consistently low misalignment from the beginning, indicating a distinct trend. We term the observed persistent or rebounding divergence between LRMs and non-reasoning models the ***Global Misalignment Rebound*** phenomenon. This phenomenon, characteristic of LRM comparisons with non-reasoning models, is mainly caused by LRMs continuously generating thinking patterns at the beginning of sentences, partly to prevent premature conclusion of the generation process. This finding demonstrates that merely extending context length is insufficient to resolve the misalignment between reasoning and non-reasoning models, indicating that reasoning capability is *not* as superficial as instruction-following.

Local Misalignment Diminish. It is uncommon that a longer context does not benefit the alignment. Thus, to further understand this behavior, we conduct the sentence-level analysis by calculating the token misalignment rate at each sentence-level position. In the response, sentences can be separated by periods, question marks, exclamation marks, and the newline symbol. Specifically, for any position x, we first collect every sentence that is at least x tokens long. Mark the x-th token in each of those sentences as 1 if it is misaligned and 0 if it is aligned. The average of these 0-1 indicators across all selected sentences is the misalignment rate for position x.

As shown in Figure 1 (lower), for the red line, there is no obvious misalignment decrease that can be observed. It means that between the instruct and the base model, the misalignment occurs relatively evenly across the whole sentence. On the contrary, for LRM-involved model combinations, the blue, orange, and green lines, the misalignment rates drop dramatically at the first several tokens, e.g., from 0.4 to 0.15, and then keep diminishing, indicating a totally different behavior. Thus, we term this phenomenon the ***Local Misalignment Diminish*** phenomenon for reasoning models. These findings reveal a novel periodical, sentence-level misalignment diminish pattern unique to LRMs, driven by thinking-pattern indicators concentrated at sentence openings, shedding light on a better understanding of token-level divergences of these two types of models.

Findings. From this section, several key findings can be concluded:

- LRM misalignment with non-reasoning models, while largely superficial and characterized by stylistic "thinking cues", uniquely exhibits a ***Global Misalignment Rebound***. Unlike instruct models that increasingly align with more context, token divergence at the response level can persist or even grow, underscoring deeper, ingrained differences in their generative behavior.
- LRMs distinctively display a ***Local Misalignment Diminish***. This manifests as a novel, periodical sentence-level pattern where high misalignment, driven by "thinking cues" concentrated at sentence beginnings, rapidly decreases as the sentence progresses. This predictable intra-sentence dynamic is a crucial insight for developing LRM-guided decoding and understanding LRM patterns.

3 FoReaL-Decoding

Motivated by the above token divergence analysis, we propose a collaborative fast-slow thinking decoding method for cost-quality Trade-off, ***Fo**llow the **Rea**soning **L**eader (FoReaL-Decoding)*, a plug-and-play training-free method that mixes the strength of a slow but highly capable large reasoning model with the speed of a small model. The central idea is to let the strong, large (*Leading*) model lead at the beginning of sentences, and allow the weaker, small (*Draft*) model to complete the rest of the tokens. This decoding algorithm is of strong controllability, which can smoothly transfer into the Leading model only or downgrade to the Draft model only, by controlling the probability and the number of tokens to lead. Details can be found in the Appendix.

4 Conclusion

Our systematic token-level analysis comparing Large Reasoning Models (LRMs) with non-reasoning models has uncovered two pivotal, previously under-explored divergence phenomena. First, we identified a ***Global Misalignment Rebound***, where LRM token divergence from non-reasoning models can unexpectedly persist or even increase over entire responses, underscoring deep-seated generative differences not easily bridged by extended context. Second, and critically for our method, we characterized the ***Local Misalignment Diminish***: a novel, periodical sentence-level pattern wherein LRM-specific stylistic "thinking cues" cause high token divergence at the very beginning of sentences, after which this misalignment rapidly decreases within the sentence.

References

[1] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025.

[2] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In Neele Falk, Sara Papi, and Mike Zhang, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta, March 2024. Association for Computational Linguistics.

[3] AI-MO. AIME 2022–2024 Validation Set, 2024.

[4] AI-MO. AMC 12 2023 Integer-Answer Validation Set, 2024.

[5] Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, Tomasz Sternal, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Łukasz Flis, Hannes Eberhard, Hubert Niewiadomski, and Torsten Hoefler. Reasoning language models: A blueprint, 2025.

[6] Alexander Bukharin and Tuo Zhao. Data diversity matters for robust instruction tuning, 2023.

[7] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[8] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[9] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca with fewer data, 2023.

[10] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025.

[11] Runjin Chen, Gabriel Jacob Perin, Xuxi Chen, Xilun Chen, Yan Han, Nina ST Hirata, Junyuan Hong, and Bhavya Kailkhura. Extracting and understanding the superficial knowledge in alignment. *arXiv preprint arXiv:2502.04602*, 2025.

[12] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025.

[13] Yushuo Chen, Tianyi Tang, Erge Xiang, Linjiang Li, Wayne Xin Zhao, Jing Wang, Yunpeng Chai, and Ji-Rong Wen. Towards coarse-to-fine evaluation of inference efficiency for large language models. *arXiv preprint arXiv:2404.11502*, 2024.

[14] Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks, 2025.

[15] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, 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.

[16] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, and etc. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[17] Qianlong Du, Chengqing Zong, and Jiajun Zhang. Mods: Model-oriented data selection for instruction tuning, 2023.

[18] Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*, 2025.

[19] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

[20] Xiaotian Han. Reproduce the inference-time scaling experiment, 2024.

[21] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[22] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey, 2023.

[23] Lifeng Jin, Baolin Peng, Linfeng Song, Haitao Mi, Ye Tian, and Dong Yu. Collaborative decoding of critical tokens for boosting factuality of large language models. *arXiv preprint arXiv:2402.17982*, 2024.

[24] Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

[25] Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. Overthink: Slowdown attacks on reasoning llms, 2025.

[26] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[27] Jianwei Li and Jung-Eun Kim. Superficial safety alignment hypothesis. *arXiv preprint arXiv:2410.10862*, 2024.

[28] Ming Li, Han Chen, Chenguang Wang, Dang Nguyen, Dianqi Li, and Tianyi Zhou. Ruler: Improving llm controllability by rule-based data recycling. *arXiv preprint arXiv:2406.15938*, 2024.

[29] Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 16189–16211, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[30] Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, and Tianyi Zhou. Reflection-tuning: Recycling data for better instruction-tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[31] Ming Li, Pei Chen, Chenguang Wang, Hongyu Zhao, Yijun Liang, Yupeng Hou, Fuxiao Liu, and Tianyi Zhou. Mosaic-it: Free compositional data augmentation improves instruction tuning. *arXiv preprint arXiv:2405.13326*, 2024.

[32] Ming Li, Yanhong Li, Ziyue Li, and Tianyi Zhou. How instruction and reasoning data shape post-training: Data quality through the lens of layer-wise gradients. *arXiv preprint arXiv:2504.10766*, 2025.

[33] Ming Li, Yanhong Li, and Tianyi Zhou. What happened in llms layers when trained for fast vs. slow thinking: A gradient perspective. *arXiv preprint arXiv:2410.23743*, 2024.

[34] Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[35] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[36] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

[37] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models, 2025.

[38] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[39] 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.

[40] Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base llms: Rethinking alignment via in-context learning. *arXiv preprint arXiv:2312.01552*, 2023.

[41] Changshu Liu, Shizhuo Dylan Zhang, Ali Reza Ibrahimzada, and Reyhaneh Jabbarvand. Codemind: A framework to challenge large language models for code reasoning, 2024.

[42] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.

[43] Xiaoxuan Liu, Cade Daniel, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Xiangxi Mo, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Optimizing speculative decoding for serving large language models using goodput. *arXiv preprint arXiv:2406.14066*, 2024.

[44] Yue Liu, Jiaying Wu, Yufei He, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, and Bryan Hooi. Efficient inference for large reasoning models: A survey, 2025.

[45] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning, 2025.

[46] 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.

[47] OpenAI. OpenAI o1 System Card, December 2024.

[48] Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond, 2025.

[49] Mohit Raghavendra, Vaskar Nath, and Sean Hendryx. Revisiting the superficial alignment hypothesis. *arXiv preprint arXiv:2410.03717*, 2024.

[50] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[51] Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE, 2024.

[52] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[53] Shannon Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. *arXiv preprint arXiv:2403.03870*, 2024.

[54] Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. Efficient reasoning with hidden thinking. *arXiv preprint arXiv:2501.19201*, 2025.

[55] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

[56] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*, 2024.

[57] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[58] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[59] Qwen Team. Qwen3, April 2025.

[60] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and etc. Llama 2: Open foundation and fine-tuned chat models, 2023.

[61] Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Dynamic self-consistency: Leveraging reasoning paths for efficient llm sampling. *arXiv preprint arXiv:2408.17017*, 2024.

[62] Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *arXiv preprint arXiv:2504.07934*, 2025.

[63] Xiyao Wang, Zhengyuan Yang, Linjie Li, Hongjin Lu, Yuancheng Xu, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Scaling inference-time search with vision value model for improved visual comprehension. *arXiv preprint arXiv:2412.03704*, 2024.

[64] Xiyao Wang, Yuhang Zhou, Xiaoyu Liu, Hongjin Lu, Yuancheng Xu, Feihong He, Jaehong Yoon, Taixi Lu, Gedas Bertasius, Mohit Bansal, et al. Mementos: A comprehensive benchmark for multimodal large language model reasoning over image sequences. *arXiv preprint arXiv:2401.10529*, 2024.

[65] Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey, 2025.

[66] 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.

[67] Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.

[68] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

[69] Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy, 2025.

[70] Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning, 2025.

[71] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

[72] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *ArXiv*, abs/2402.13116, 2024.

[73] Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time, 2025.

[74] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025.

[75] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

[76] Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

[77] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

Table of Contents for Appendix

A FoReaL-Decoding **11**

B Experiments **12**

C Pseudo Code **15**

D FLOPs Calculation **15**

E Related Works **16**

F Detailed Results **18**

A FoReaL-Decoding

Motivated by the above token divergence analysis, we propose a collaborative fast-slow thinking decoding method for cost-quality Trade-off, ***Follow the Reasoning Leader (FoReaL-Decoding)***, a plug-and-play training-free method that mixes the strength of a slow but highly capable large reasoning model with the speed of a small model. The central idea is to let the strong, large (*Leading*) model lead at the beginning of sentences, and allow the weaker, small (*Draft*) model to complete the rest of the tokens. This decoding algorithm is of strong controllability, which can smoothly transfer into the Leading model only or downgrade to the Draft model only, by controlling the probability and the number of tokens to lead.

Preliminaries. The two control knobs that govern the trade-off between cost and quality:

1. **Required lead count** $n \in \mathbb{N}$: the minimum number of tokens the Leading model generates before yielding control to the Draft model.
2. **Lead probability** $p \in [0, 1]$: probability that a sentence is led by the Leading model.

When $p = 0$, the decoding system degenerates to pure Draft model decoding; when $p = 1$ and n exceeds the sentence length, it transfers to Leading model decoding. Intermediate settings form a continuity of compute–accuracy trade-offs.

In addition, let $t \in \mathbb{N}$ represent the global token index in the response, and $s \in \mathbb{N}$ represent the sentence index. $g_s \sim \mathrm{Bernoulli}(p)$ represents the sentence-level gate to decide what model to start the sentence s: the sentence will be led by the Leading model if $g_s = 1$. τ_s represents the global position of the first token in s. $s(t) = \max\{s : \tau_s \leq t\}$ is the function that maps the token t to the sentence index that t belongs to. $\lambda_t = t - \tau_{s(t)} + 1$ is the local position of token t within its sentence.

Intra-Sentence Lead Within a sentence s, the generation of each token at position t is governed by the token-level policy,

$$\pi_t \;=\; \begin{cases} L & g_{s(t)} = 1 \;\wedge\; \big[\lambda_t \leq n \;\vee\; t < H^{\mathrm{hit}}_{s(t)}\big], \\ D & \text{otherwise.} \end{cases} \tag{1}$$

$g_{s(t)} = 1$ represents this sentence $s(t)$ should be led by the Leading model, decided by the gate. L and D represent the Leading model and Draft model, respectively. $\lambda_t \leq n$ represents the index of this token within this sentence that is smaller than the required lead count n, thus should be generated by the Leading model. H^{hit}_s is the first token index within s where the top-1 token generated by the Draft model matches that of the Leading model for k consecutive steps:

$$H^{\mathrm{hit}}_s = \min\{t : \, s(t) = s, \, \lambda_t > n, \, h_t = k\}, \tag{2}$$

where h_t represents the number of consecutive hits within the max sliding window of k:

$$h_t = \sum_{i=0}^{k-1} \delta_{t-i}, \quad \delta_t = \mathbf{1}\big\{\arg\max_{y \in \mathcal{V}} P_D(\cdot|c_t) = \arg\max_{y \in \mathcal{V}} P_L(\cdot|c_t)\big\} \tag{3}$$

Put it simply, for each sentence, if the Bernoulli gate decides to let P_L lead the sentence with the probability p, P_L will generate the first n tokens. Then, P_D begins the generation process as well, with the purpose of measuring the alignment between the two models. When the top-1 predictions of these two models aligned with each other for k times, the generation process is yielded to P_D, otherwise, P_L generates the whole sentence. On the contrary, if the gate decides not to let P_L lead, then the whole sentence will be completely generated by P_D.

Sentence-level likelihood. For sentence s with token span $Y_s = (y_{\tau_s}, \ldots, y_{\tau_{s+1}-1})$ and length L_s, the conditional likelihood under FoReaL-Decoding is:

$$P_{\mathrm{CoL}}\big(Y_s \mid g_s\big) \;=\; \prod_{i=0}^{L_s-1} P_{\pi_{\tau_s+i}}\big(y_{\tau_s+i} \mid c_{\tau_s+i}\big), \tag{4}$$

Table 1: Comparisons of Accuracy and Efficiency (TFLOPs) of FoReaL-Decoding on commonly used reasoning-heavy math problem tasks. To further show the wide trade-off scopes of our method, we provide some different configurations as the control. The results of Speculative Thinking are the reported results. The accuracies are better with higher (↑) values, while the TFLOPs are better with lower (↓) values. The accuracies on each line are compared with the Draft model, and the TFLOPs are compared with the Leading models: better values are colored in green, otherwise red.

Model		AIME24		GPQA-D		MATH500		AMC23	
Method	**Config**	**ACC (%) ↑**	**TFLOPs ↓**	**ACC (%) ↑**	**TFLOPs ↓**	**ACC (%) ↑**	**TFLOPs ↓**	**ACC (%) ↑**	**TFLOPs ↓**
DeepSeek-R1-Distill-Qwen-32B + DeepSeek-R1-Distill-Qwen-1.5B									
DeepSeek-R1-Distill-Qwen-32B		66.7	15.72	59.6	8.09	93.6	4.13	95.0	7.54
DeepSeek-R1-Distill-Qwen-1.5B		23.3	2.86	22.2	1.13	81.4	1.14	65.0	2.51
Speculative Thinking		32.2	5.75	41.9	2.62	89.4	1.51	80.0	3.31
FoReaL-Decoding	n=15,p=0.4	33.3 (+10.0)	**5.60 (-10.12)**	43.3 (+21.1)	**2.47 (-5.62)**	90.2 (+8.8)	**1.43 (-2.88)**	80.0 (+15.0)	**2.91 (-4.63)**
FoReaL-Decoding	n=15,p=0.6	50.0 (+26.7)	6.77 (-8.95)	48.2 (+26.0)	4.50 (-3.59)	91.4 (+10.0)	2.40 (-1.26)	80.0 (+15.0)	3.99 (-3.55)
FoReaL-Decoding	n=15,p=0.8	50.0 (+26.7)	8.47 (-7.25)	54.6 (+32.4)	4.69 (-3.40)	93.4 (+12.0)	2.70 (-1.43)	90.0 (+25.0)	5.37 (-2.17)
FoReaL-Decoding	n=15,p=1.0	**66.7 (+43.4)**	9.16 (-6.56)	56.6 (+34.4)	6.21 (-1.88)	93.2 (+11.8)	3.14 (-0.99)	92.5 (+27.5)	5.28 (-2.26)
FoReaL-Decoding	n=25,p=0.8	53.3 (+30.0)	10.95 (-4.77)	57.1 (+34.9)	5.65 (-2.44)	92.6 (+11.2)	3.13 (-1.00)	92.5 (+27.5)	4.99 (-2.55)
FoReaL-Decoding	n=25,p=1.0	**66.7 (+43.4)**	10.54 (-5.18)	**57.6 (+35.4)**	6.68 (-1.41)	**94.5 (+13.1)**	3.50 (-0.63)	**95.0 (+30.0)**	5.66 (-1.88)
DeepSeek-R1-Distill-Qwen-32B + Qwen2.5-1.5B-Instruct									
DeepSeek-R1-Distill-Qwen-32B		66.7	15.72	59.6	8.09	93.6	4.13	95.0	7.54
Qwen2.5-1.5B-Instruct		0.0	0.12	23.7	0.12	49.2	0.09	15.0	0.10
FoReaL-Decoding	n=15,p=0.8	20.0 (+20.0)	**9.05 (-6.67)**	38.4 (+14.7)	5.63 (-2.46)	76.2 (+27.0)	2.85 (-1.28)	65.0 (+50.0)	5.22 (-2.32)
FoReaL-Decoding	n=15,p=1.0	20.0 (+20.0)	11.19 (-4.53)	47.5 (+23.8)	5.86 (-2.23)	85.9 (+36.7)	3.28 (-0.85)	85.0 (-70.0)	6.15 (-1.39)
FoReaL-Decoding	n=25,p=0.8	36.7 (+36.7)	9.58 (-6.14)	45.0 (+21.3)	**4.37 (-3.72)**	82.0 (+32.8)	**2.52 (-1.61)**	72.5 (+57.5)	**4.65 (-2.89)**
FoReaL-Decoding	n=25,p=1.0	**40.0 (+40.0)**	11.00 (-4.72)	**57.1 (+33.4)**	6.27 (-1.82)	**90.8 (+2.8)**	3.36 (-0.77)	**92.5 (-77.5)**	6.88 (-0.66)
DeepSeek-R1-Distill-Qwen-1.5B + Qwen2.5-7B-Instruct									
DeepSeek-R1-Distill-Qwen-1.5B		23.3	2.86	22.2	1.13	81.4	1.14	65.0	2.51
Qwen2.5-7B-Instruct		6.7	0.95	38.4	0.89	76.0	0.61	52.5	0.75
Speculative Thinking		6.7	4.93	31.8	6.73	74.8	2.04	55.0	4.97
FoReaL-Decoding	n=15,p=0.8	16.7 (+10.0)	2.05 (-0.81)	**34.3 (-4.1)**	1.07 (-0.06)	76.4 (+0.4)	0.57 (-0.57)	57.5 (+5.0)	**1.08 (-1.43)**
FoReaL-Decoding	n=15,p=1.0	16.7 (+10.0)	6.47 (+3.61)	29.8 (-8.6)	3.08 (+1.95)	**79.6 (+3.6)**	1.42 (+0.28)	52.5 (+0.0)	3.35 (+0.84)
FoReaL-Decoding	n=25,p=0.8	20.0 (+13.3)	**1.57 (-1.29)**	33.3 (-5.1)	**0.80 (-0.33)**	78.6 (+2.6)	**0.55 (-0.59)**	**65.0 (+12.5)**	1.76 (-0.75)
FoReaL-Decoding	n=25,p=1.0	**23.3 (+16.6)**	3.18 (+0.32)	29.3 (-9.1)	2.53 (+1.40)	79.2 (+3.2)	1.04 (-0.10)	**65.0 (+12.5)**	1.66 (-0.85)

Whenever $\pi_t = L$, the factor draws its probability from the distribution P_L of the Leader model; otherwise from the Draft model of distribution P_D.

Inter-Sentence Transfer Upon encountering a sentence boundary at the token t, i.e., the sentence is complete, we execute the inter-sentence update by resetting the hit counter and resampling the gate for the next sentence.

$$s \leftarrow s + 1, \quad g_s \sim \text{Bernoulli}(p), \quad h_t \leftarrow 0 \tag{5}$$

Comparisons with related methods. In speculative decoding [26], the final text provably matches what the large model alone would have produced. However, our ***FoReaL-Decoding*** focuses on the reasoning-heavy scenarios where the responses generated by the LRM itself are not desirable due to the overthinking. Thus, our method serves as a deliberate mixture of two distributions, aiming at reducing the overthinking problem of LRMs by inserting the distribution from weaker models, and at the same time increasing the efficiency. A recent work, RSD [38], also aims at reducing computation cost by utilizing speculative decoding. However, it introduces an additional process reward model as the judge, while our method focuses on utilizing the collaborative models themselves only, thus, it is largely different from our settings. Another concurrent work, Speculative Thinking [73], also shares similar motivation as ours, in which a "small-writes, large-fixes" mechanism is utilized, which differs from our "large-leads, small-follows". Moreover, ***FoReaL-Decoding*** provides a smooth transition from the small to the large model, representing wider trade-off scopes.

B Experiments

B.1 Implementation Details

Models, Datasets, and Setup. To assess the effects of FoReaL-Decoding, extensive experiments are conducted for different model combinations in the Qwen2.5 family, including reasoning models like *R1-Distill-Qwen-32B* [16], *R1-Distill-Qwen-1.5B* [16], non-reasoning instruct models like *Qwen2.5-7B-Instruct* [58], *Qwen2.5-1.5B-Instruct* [58], and base models like *Qwen2.5-1.5B* [58]. To cover a wide scope of potential trade-offs, we utilize the reasoning models as the Leading models, while any of the above types as the Draft models. Moreover, our extensive experiments on the recently

released *Qwen3* [59] series further verify the generalizability of our method. We evaluate our method on relatively hard, reasoning-heavy math datasets, including AIME2024 [3], GPQA-Diamond [50], AMC23 [4], and MATH500 [39]. All experiments were conducted on NVIDIA A100 GPUs (80G), utilizing the Huggingface Transformers package. During the generation, we follow the recommended generation configuration from R1-Distill models as `temperature`=0.6, `top_p`=0.95, `top_k`=40 for all the experiments. During the generation, we always let the Leading model generate the first paragraph, and we fix the required hits for generation transfer as $k = 5$ for all the experiments.

B.2 Main Results

Table 1 presents the comparisons between accuracy and efficiency (TFLOPs) of FoReaL-Decoding on commonly used reasoning-heavy math problem tasks. We provide some different configurations as controls to show the wide trade-off scopes of our method. We also present the reported results of the concurrent work, Speculative Thinking [73], for better comparison. The accuracies on each line are compared with the Draft model, and the TFLOPs are compared with the Leading models: better values are colored in green, otherwise red. We utilize the theoretically estimated TFLOPs as the efficiency measurement since it takes the generation length into account, different from the estimated speed. In the main comparison, we focus on 3 collaborative settings. Across four benchmarks, FoReaL-Decoding cuts inference cost by 30 – 55% relative to Leader-only decoding while retaining 86 – 100% of its accuracy. The detailed statistics, including response length and leading ratios on AIME24, can be found in Table 2 for better understanding.

R1-Distill-Qwen-32B for Leading, R1-Distill-Qwen-1.5B for Draft. This collaborative setting yields the highest accuracies for all of the math reasoning datasets. In this setting, the larger 32B reasoning model takes charge of the leading of the sentences, while the smaller 1.5B reasoning model needs to complete the remaining sentence. In this setting, both models have the reasoning capabilities, but FoReaL-Decoding implicitly separates the generation of each sentence into two phases and yields the less informative Draft phase to the smaller model for better efficiency. As shown in the table, all our results obtain better performances compared with the Draft model and efficiencies compared with the Leading model, and also exceed Speculative Thinking, indicating the capability of our methods. Moreover, on all of the tasks except GPQA-D, FoReaL-Decoding reaches similar or even slightly higher performances than the 32B Leading model with fewer TFLOPs, (-6.56 on AIME24, -0.63 on MATH500, and -1.88 on AMC).

Table 2: The detailed results of different collaborative settings on AIME. Length represents the averaged response length, Ratio represents the average ratio of tokens decoded by using the Leading model, for each task. Additional configuration that uses base model for Draft is included.

Model		AIME24			
Method	**Config**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**
DeepSeek-R1-Distill-Qwen-32B + DeepSeek-R1-Distill-Qwen-1.5B					
FoReaL-Decoding	n=15, p=0.4	33.3	11 876	0.272	5.60
FoReaL-Decoding	n=15, p=0.6	50.0	10 934	0.401	6.77
FoReaL-Decoding	n=15, p=0.8	50.0	11 532	0.527	8.47
FoReaL-Decoding	n=15, p=1.0	66.7	10 617	0.666	9.16
FoReaL-Decoding	n=25, p=0.8	53.3	12 081	0.676	10.95
FoReaL-Decoding	n=25, p=1.0	66.7	11 116	0.683	10.54
DeepSeek-R1-Distill-Qwen-32B + Qwen2.5-1.5B-Instruct					
FoReaL-Decoding	n=15, p=0.8	20.0	12 584	0.571	9.05
FoReaL-Decoding	n=15, p=1.0	20.0	14 188	0.588	11.19
FoReaL-Decoding	n=25, p=0.8	36.7	11 575	0.710	9.58
FoReaL-Decoding	n=25, p=1.0	40.0	11 239	0.813	11.00
DeepSeek-R1-Distill-Qwen-32B + Qwen2.5-1.5B (Base)					
FoReaL-Decoding	n=15, p=0.8	23.3	12 224	0.547	9.56
FoReaL-Decoding	n=15, p=1.0	20.0	12 107	0.664	10.39
DeepSeek-R1-Distill-Qwen-1.5B + Qwen2.5-7B-Instruct					
FoReaL-Decoding	n=15, p=0.8	16.7	4 120	0.545	2.05
FoReaL-Decoding	n=15, p=1.0	16.7	14 132	0.651	6.47
FoReaL-Decoding	n=25, p=0.8	20.0	4 474	0.693	1.57
FoReaL-Decoding	n=25, p=1.0	23.3	11 436	0.841	3.18

R1-Distill-Qwen-32B for Leading, Qwen2.5-1.5B-Instruct for Draft. This setting represents a direct mixture of a large reasoning model and a small non-reasoning model. As shown in the table, the 1.5B instruct model performs badly on the given difficult math problems. The use of a stronger reasoning model for leading largely improves the accuracy, although with more computation required. The response lengths are largely shorter than *R1-Distill-Qwen-1.5B*, representing an alleviation of overthinking. Compared with using another small reasoning model for Draft, utilizing the instruction model leads to suboptimal performance. To understand this phenomenon, further experiments are conducted where the base pretrained model *Qwen2.5-1.5B* is utilized as the Draft model. As shown in Table 2, the accuracies, response lengths, and TFLOPs are almost identical compared with using base and instruct models, which means the previous instruction-aligned process does not benefit the current reasoning settings.

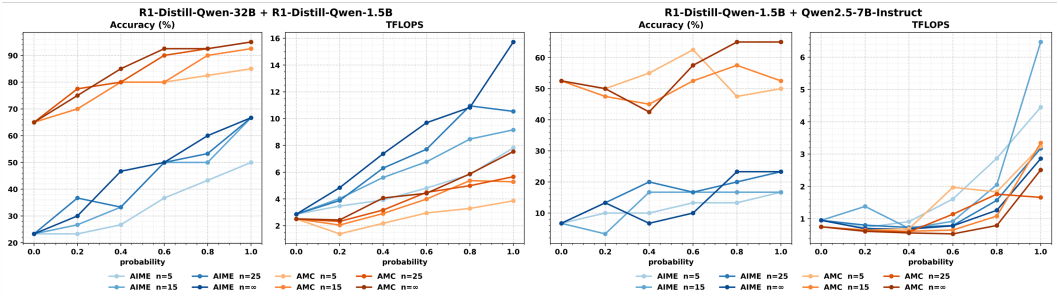

Figure 4: Effects of lead count and lead probability on AIME24 and AMC23 datasets, based on 2 collaborative configurations. FoReaL-Decoding provides a smooth cost-quality trade-off, making the transition from the weak Draft model to the strong Leading model smooth and controllable.

R1-Distill-Qwen-1.5B for Leading, Qwen2.5-7B-Instruct for Draft. Different from the above settings, in which a strong but large reasoning model is used as the Leading model, this setting considers a different and most efficient scenario, utilizing a small reasoning model for leading and a slightly larger instruct model for Draft. In this setting, the efficiencies are reduced to an extremely low level, even faster than directly utilizing the small reasoning models. As shown in Table 2, FoReaL-Decoding largely reduces the length required for the problem, thus largely reducing the computation required. On AIME24 and AMC23, our method reaches the same accuracy as the Leading model with similar or less computation. On GPQA, our method reaches an intermediate accuracy, since the abnormal situation where a non-reasoning model has better performance than the reasoning model.

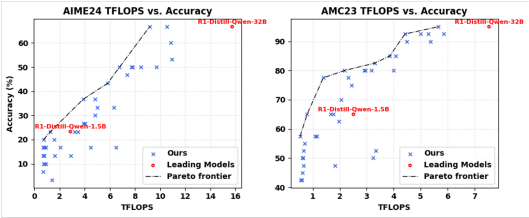

Figure 3: The trade-off curves between accuracy and TFLOPs. Blue markers correspond to FoReaL-Decoding variants, red circles denote the corresponding LRMs, and the dashed line is the empirically computed Pareto frontier. On both benchmarks, every LRM point is Pareto dominated.

Estimation of TFLOPs. Empirical latency depends on vendor-specific kernel fusion and memory layouts, so a timing measured on one backend may not transfer to another. Counting floating-point operations (FLOPs) provides a hardware-agnostic yardstick that isolates algorithmic differences. The performance figures we report are presented in TeraFLOPs (TFLOPs), where one TFLOP equals 10^{12} FLOPs. Typically, the generation process proceeds in two modes, prefill and decode. Prefill processes the full prompt of length s once without any KV cache, and decode autoregressively emits output tokens while re-using cached keys and values. When GPU memory is sufficient, profiling shows that producing multiple tokens during the prefix phase costs almost the same as decoding a single token. Therefore, we upper-bound the prefix cost by the single-token decode cost. For the TFLOPs values cited in our results, we calculate the precise total FLOPs using the detailed formulas presented in Appendix A.1. This calculation methodology is based on [13, 20], and the resulting total FLOPs are then converted to TFLOPs for reporting.

B.3 Effects of Lead Count and Lead Probability

Figure 4 sweeps the two hyperparameters that govern the controllability of FoReaL-Decoding, lead count n and lead probability p on AIME24 and AMC23 datasets, based on 2 collaborative configurations, *DeepSeek-R1-Distill-Qwen-32B* + *DeepSeek-R1-Distill-Qwen-1.5B* and *DeepSeek-R1-Distill-Qwen-1.5B* + *Qwen2.5-7B-Instruct*, representing the high-performance and high-efficiency settings, respectively. For each model combination, we run experiments on $n \in \{5, 15, 25, +\infty\}, p \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. When $p = 0$, FoReaL-Decoding utilizes the Draft model only, and utilizes the Leading model only when $p = 1$ and $n = +\infty$. According to the figure, FoReaL-Decoding provides a smooth cost-quality trade-off, making the transition from the weak Draft model to the strong Leading model smooth and controllable. For any fixed n, increasing the probability p of the Leader intervention shifts the operating point up and to the right: accuracy rises while TFLOPs grow almost linearly. The resulting curve is smooth, allowing practitioners to trade latency for quality by adjusting (n, p). The jump from $n = 5$ to $n = 15$ yields large accuracy gains at a modest cost

increase. Further enlarging the Leader count ($n \geq 25$) adds little accuracy yet inflates compute up a lot, indicating that sentence-level guidance already captures most of the benefit of slow reasoning. A sweet spot is around $(n, p) = (15, 0.6)$.

B.4 Trade-Off Curves

Figure 3 plots the trade-off curves between accuracy and TFLOPs for every (n, p) configuration tested on AIME24 (left) and AMC23 (right), according to the experiment scopes from the above section. Blue markers correspond to FoReaL-Decoding variants, red circles denote the corresponding LRMs, and the dashed line is the empirically computed Pareto frontier. On both benchmarks, every LRM point is Pareto dominated: an alternative FoReaL-Decoding setting always achieves higher accuracy at lower cost. Moreover, we find that the frontier rises sharply between 0.5 and 2 TFLOPs, as each additional TFLOP yields 10–15 percentage points of accuracy. However, beyond ≈5 TFLOPs, the curve flattens; extra compute buys only marginal improvements up to the ceiling.

B.5 Results on Qwen3 Families

To further verify the effectiveness and generalizability of FoReaL-Decoding, additional experiments are conducted on the Qwen3 series of models, including *Qwen3-32B*, *Qwen3-1.7B*, and *Qwen3-0.6B*, due to the various sizes of models provided in the family. Specifically, we utilize the reasoning modes for these models and follow exactly the same generation configuration for our main experiments. The detailed experimental results are shown in Table 3. As shown in the table, FoReaL-Decoding shows promising performances on both *Qwen3-32B* + *Qwen3-1.7B* and *Qwen3-32B* + *Qwen3-0.6B* configurations. For the former configuration, FoReaL-Decoding reaches a similar accuracy (73.3% to 76.6%) with approximately half of the TFLOPs (8.83 to 15.75).

Table 3: The detailed results of Qwen3 series models on AIME. FoReaL-Decoding shows promising performance in this additional family.

Model		AIME24			
Method	**Config**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**
Base Models					
Qwen3-32B	–	76.6	13 275	–	15.75
Qwen3-1.7B	–	40.0	14 990	–	2.81
Qwen3-0.6B	–	13.3	15 839	–	1.14
Qwen3-32B + Qwen3-1.7B					
FoReaL-Decoding	n=15, p=0.4	60.0	14 840	0.272	7.20
FoReaL-Decoding	n=15, p=0.6	73.3	14 110	0.412	8.83
FoReaL-Decoding	n=15, p=0.8	73.3	15 081	0.536	11.43
Qwen3-32B + Qwen3-0.6B					
FoReaL-Decoding	n=15, p=0.4	36.7	17 782	0.281	7.44
FoReaL-Decoding	n=15, p=0.6	63.0	14 279	0.410	8.18
FoReaL-Decoding	n=15, p=0.8	60.0	15 478	0.560	11.01

C Pseudo Code

The pseudo code of our FoReaL-Decoding is provided below, all the variables are kept the same as in the main context.

D FLOPs Calculation

The calculation of Floating Point Operations (FLOPs) for the prefill and decoding stages is based on the methodology from [13, 20]. These calculations assume a batch size of 1.

The variables involved are defined as:

- s: Represents the sequence length.
  - For the prefill stage ($\text{FLOPs}_{\text{prefill}}(s)$), s is the length of the input prompt, denoted as p_l.
  - For the decode stage ($\text{FLOPs}_{\text{decode}}(s)$), s is the current length of the context (prompt + tokens generated so far) that the model attends to via its Key-Value (KV) cache.
- h: The hidden size of the model.
- h': The intermediate size of the feed-forward network (FFN).
- n: The number of attention heads.
- p_l: The length of the initial problem prompt.
- d_l: The number of tokens to be generated in the solution.

It is noted that the hidden size h relates to the number of attention heads n and the size of each attention head d by $h = n \cdot d$.

Algorithm 1: FoReaL-Decoding

Input: Leading model P_L, Draft model P_D, lead count n, lead probability p, hit threshold k, input prompt q, max new tokens MAX_LEN
Output: Generated tokens $\mathbf{y}$

```
y ← [], h ← 0, λ ← 0;
c ← q ;                                                  // Initial context
g ← 1 ;                                                  // Initialize gate
while len(y) < MAX_LEN do
    if is_sentence_boundary(y[−1]) then
        g ∼ Bernoulli(p) ;                               // Sample gate for new sentence
        h ← 0 ;                                          // Reset hit counter
        λ ← 0 ;                                          // Reset position in sentence
    λ ← λ + 1 ;                                          // Increment position in sentence
    // Generate next token
    if g = 1 and (λ ≤ n or h < k) then
        t ← sample(P_L(·|c)) ;                           // Use Leading model
    else
        t ← sample(P_D(·|c)) ;                           // Use Draft model
    // Check alignment when approaching transition point
    if g = 1 and λ > n − k then
        if top-1(P_D(·|c)) = top-1(P_L(·|c)) then
            h ← h + 1;
        else
            h ← 0;
    y.append(t);
    c ← concat(c, t) ;                                   // Update context
    if t ∈ EOS_tokens then
        break;
return y;
```

The FLOPs for the prefill stage, which processes the initial input prompt of length $s = p_l$, is given by Equation 6:

$$\text{FLOPs}_{\text{prefill}}(s) = 8sh^2 + 16sh + 4s^2h + 4s^2n + 6shh' + 2sh' \tag{6}$$

The FLOPs for the decode stage, which generates a single token when the current KV cache has a length of s, is given by Equation 7:

$$\text{FLOPs}_{\text{decode}}(s) = 8h^2 + 16h + 4sh + 4sn + 6hh' + 2h' \tag{7}$$

The total FLOPs to generate d_l tokens from a prompt of length p_l combines the prefill cost for the prompt and the sum of decode costs for each generated token, as shown in Equation 8:

$$\text{FLOPs}_{\text{total}} = \text{FLOPs}_{\text{prefill}}(p_l) + \sum_{i=0}^{d_l-1} \text{FLOPs}_{\text{decode}}(p_l + i) \tag{8}$$

In this formula, for the i-th token being generated (0-indexed), the argument to $\text{FLOPs}_{\text{decode}}$ is $p_l + i$, representing the sequence length in the KV cache at that generation step.

E Related Works

E.1 Large Reasoning Models

Recent advances in large language models (LLMs) have spurred a surge of work aimed at strengthening their reasoning abilities [2, 5, 10]. Core reasoning skills are already instilled during pre-training, where models absorb commonsense and mathematical patterns from vast text corpora [60, 47]. Researchers have therefore concentrated on post-training techniques to further polish these skills. One

prominent direction employs reinforcement learning to nudge models toward more effective chains of thought [52, 70, 15, 62]. Another line shows that carefully curated instruction-tuning data can likewise deliver sizable gains in reasoning accuracy [74, 46, 63].

Despite the impressive benchmark scores of recent Reasoning Language Models, several studies have begun to probe the quality and efficiency of the reasoning they generate. [69] conduct a broad assessment and reveal substantial redundancy in many model-produced solutions. Follow-up investigations [12, 14, 48, 44, 18] underscore an "overthinking" phenomenon, whereby models craft unduly verbose derivations even for simple problems. Capitalizing on this trait, [25] demonstrate a slowdown attack: small input perturbations can trigger excessive reasoning, markedly degrading inference speed.

To alleviate overthinking and improve efficiency for reasoning models, a series of efficient reasoning methods has been proposed. For example, [75, 57, 1, 68, 45] utilize model-based methods that either add further constraints on RL rewards or SFT on diverse lengths of CoTs, [21, 55, 54, 76] utilize latent-space reasoning methods that transfer the massive tokens into the embedding space, [19, 71, 51] utilize the prompt-based methods, [56, 61, 67] utilize the sampling methods. Most of these methods either require further post-training or manipulating the distribution of LRM itself.

E.2 Alignment and Token Pattern Analysis

A key empirical foundation for LLM Alignment is LIMA [77], which demonstrated that just $1,000$ carefully curated instruction–response pairs are already enough for LLM alignment, crystallizing the "superficial alignment" hypothesis. While a line of work directly follows the hypotheses by introducing data selection or alignment methods [9, 35, 30, 29, 17, 28, 6, 42, 34, 31, 32, 72], there are also works that try to further investigate this phenomenon.

[40] provides a comprehensive token-level evidence by comparing the top-k token distributions of base models and their chat-tuned counterparts. The authors show that almost all divergence concentrates on discourse markers, politeness phrases, and safety disclaimers, while core content tokens remain unchanged. [11] dissects which prompt-level cues are sufficient (and which are not) for alignment, showing that reasoning gaps emerge precisely where superficial patterns end. The debate has sparked push-back as well: [49] demonstrates systematic performance gains when the amount of post-training data scales up, arguing that some deeper representational changes do accrue beyond mere style. Researchers are also probing where superficial signals live: [27] argues that data curation, not extra optimization steps, is the primary lever: filtering for safety disclaimers yields larger alignment jumps than adding thousands of generic examples. Together, these works paint a nuanced picture: much of the alignment gap after pre-training is indeed "superficial", residing in a narrow band of stylistic tokens that can be manipulated through tiny prompts, judicious data selection. However, in this paper, we show that *the reasoning capabilities might not be as superficial as previous findings.*

E.3 Speculative Decoding and Collaborative Decoding

Speculative decoding, inaugurated by [26], uses a small "draft" model to propose several tokens that the large "target" model then verifies in one batch, yielding 2–$3\times$ latency reductions with provably identical output distributions. Follow-up work, such as [8] extends the idea to 70 B-parameter models and confirms similar speed-ups, while [7] replaces the external draft model with extra decoding heads to remove system complexity System-level schedulers like [43] dynamically adapt draft length to traffic conditions and push end-to-end gains beyond $3\times$ in production settings.

Collaborative decoding improves text quality by letting multiple models cooperate during generation. [36] runs a weak "amateur" model alongside a strong "expert" and selects tokens that maximize their likelihood gap, sharply reducing repetition and incoherence without retraining. [23] introduces a critical-token strategy that switches to the pretrained base model whenever factual precision is needed, cutting hallucinations in instruction-tuned LLMs. At an even finer grain, [53] treats "who should emit the next token" as a latent variable, enabling on-the-fly delegation between a generalist LLM and domain specialists and outperforming any single model on cross-domain tasks.

For the recent models with strong reasoning capabilities, several recent works [38, 73] based on speculative decoding have also been released, which we have introduced in the main method section. Our FoReaL-Decoding provides a different method with a better trade-off scope.

F Detailed Results

Table 4 and Table 5 show the detailed results of different settings of our method.

Table 4: The detailed results of different collaborative settings on AIME24, GPQA-D, MATH500, and AMC23, including length and ratio.

Model		AIME24				GPQA-D				MATH500				AMC23			
Method	**Config**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**
DeepSeek-R1-Distill-Qwen-32B + DeepSeek-R1-Distill-Qwen-1.5B																	
DeepSeek-R1-Distill-Qwen-32B		66.7	13035	-	15.72	59.6	6602	-	8.09	93.6	3542	-	4.13	95.0	6243	-	7.54
DeepSeek-R1-Distill-Qwen-1.5B		23.3	18021	-	2.86	22.2	8696	-	1.13	81.4	6704	-	1.14	65.0	13311	-	2.51
FoReaL-Decoding	n=15, p=0.4	33.3	11876	0.272	5.60	43.3	5841	0.294	2.47	90.2	3402	0.312	1.45	80.0	6043	0.304	2.91
FoReaL-Decoding	n=15, p=0.6	50.0	10934	0.401	6.77	48.2	7007	0.431	4.50	91.4	3995	0.452	2.40	80.0	6460	0.429	3.99
FoReaL-Decoding	n=15, p=0.8	50.0	11532	0.527	8.47	54.6	6110	0.570	4.69	93.4	3658	0.590	2.70	90.0	7037	0.571	5.37
FoReaL-Decoding	n=15, p=1.0	66.7	10617	0.666	9.16	56.6	6796	0.692	6.21	93.2	3655	0.726	3.14	92.5	5942	0.708	5.28
FoReaL-Decoding	n=25, p=0.8	53.3	12081	0.676	10.95	57.7	6223	0.702	5.65	92.6	3585	0.719	3.13	92.5	5529	0.710	4.99
FoReaL-Decoding	n=25, p=1.0	66.7	11116	0.683	10.54	57.6	6065	0.882	6.68	94.5	3403	0.890	3.50	95.0	5422	0.872	5.66
DeepSeek-R1-Distill-Qwen-32B + Qwen2.5-1.5B-Instruct																	
DeepSeek-R1-Distill-Qwen-32B		66.7	13035	-	15.72	59.6	6602	-	8.09	93.6	3542	-	4.13	95.0	6243	-	7.54
Qwen2.5-1.5B-Instruct		0.0	998	-	0.12	23.7	923	-	0.12	49.2	747	-	0.09	15.0	818	-	0.10
FoReaL-Decoding	n=15, p=0.8	20.0	12584	0.571	9.05	47.5	7013	0.587	5.63	76.2	3792	0.614	2.85	65.0	7629	0.514	5.22
FoReaL-Decoding	n=15, p=1.0	20.0	14188	0.588	11.19	47.5	6294	0.737	5.86	85.9	3894	0.750	3.28	65.0	7673	0.707	6.15
FoReaL-Decoding	n=25, p=0.8	36.7	11575	0.710	9.58	56.7	4718	0.719	4.37	82.0	3025	0.729	2.52	72.5	5415	0.649	4.65
FoReaL-Decoding	n=25, p=1.0	40.0	11239	0.813	11.00	57.1	5944	0.887	6.27	90.8	3403	0.894	3.36	92.5	6989	0.867	6.88
DeepSeek-R1-Distill-Qwen-1.5B + Qwen2.5-7B-Instruct																	
DeepSeek-R1-Distill-Qwen-1.5B		23.3	18021	-	2.86	22.2	8696	-	1.13	81.4	6704	-	1.14	65.0	13311	-	2.51
Qwen2.5-7B-Instruct		6.7	1243	-	0.95	38.4	1054	-	0.89	76.0	773	-	0.61	52.5	994	-	0.75
FoReaL-Decoding	n=15, p=0.8	16.7	4120	0.545	2.05	34.3	2130	0.602	1.07	76.4	1341	0.634	0.57	57.5	2515	0.580	1.08
FoReaL-Decoding	n=15, p=1.0	16.7	14132	0.651	6.47	29.8	7913	0.703	3.08	79.6	3480	0.735	1.42	52.5	7330	0.686	3.35
FoReaL-Decoding	n=25, p=0.8	20.0	4474	0.693	1.57	33.1	1801	0.718	0.80	78.6	1498	0.736	0.55	65.0	3778	0.683	1.76
FoReaL-Decoding	n=25, p=1.0	23.3	11436	0.841	3.18	29.3	6800	0.863	2.53	79.2	3586	0.891	1.04	60.0	5721	0.865	1.66

Table 5: The detailed results of different collaborative settings on AIME24 and AMC23, including length and ratio.

Model		AIME24				AMC23			
Method	**Config**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**	**ACC (%)**	**Length**	**Ratio**	**TFLOPs**
DeepSeek-R1-Distill-Qwen-32B + DeepSeek-R1-Distill-Qwen-1.5B									
DeepSeek-R1-Distill-Qwen-32B		66.7	13035	-	15.72	95.0	6243	-	7.54
DeepSeek-R1-Distill-Qwen-1.5B		23.3	18021	-	2.86	65.0	13311	-	2.51
FoReaL-Decoding	n=5, p=0.2	23.3	12926	0.076	3.47	77.5	5634	0.089	1.39
FoReaL-Decoding	n=5, p=0.4	26.7	11590	0.145	3.92	80.0	6549	0.157	2.18
FoReaL-Decoding	n=5, p=0.6	36.7	11560	0.202	4.81	80.0	7081	0.228	2.95
FoReaL-Decoding	n=5, p=0.8	43.3	11907	0.270	5.82	82.5	6399	0.294	3.29
FoReaL-Decoding	n=5, p=1.0	50.0	13750	0.328	7.82	85.0	6916	0.355	3.86
FoReaL-Decoding	n=15, p=0.2	26.7	12457	0.138	4.03	70.0	6680	0.154	2.05
FoReaL-Decoding	n=15, p=0.4	33.3	11876	0.272	5.60	80.0	6043	0.303	2.91
FoReaL-Decoding	n=15, p=0.6	50.0	10934	0.401	6.77	80.0	6460	0.429	3.99
FoReaL-Decoding	n=15, p=0.8	50.0	11532	0.527	8.47	90.0	7037	0.571	5.37
FoReaL-Decoding	n=15, p=1.0	66.7	10617	0.666	9.16	92.5	5942	0.708	5.28
FoReaL-Decoding	n=25, p=0.2	36.7	10805	0.178	3.88	77.5	6798	0.193	2.32
FoReaL-Decoding	n=25, p=0.4	33.3	11428	0.347	6.30	80.0	5929	0.362	3.17
FoReaL-Decoding	n=25, p=0.6	50.0	10816	0.515	7.71	90.0	6169	0.537	4.49
FoReaL-Decoding	n=25, p=0.8	53.3	12081	0.675	10.95	92.5	5529	0.710	4.99
FoReaL-Decoding	n=25, p=1.0	66.7	11117	0.683	10.54	95.0	5422	0.872	5.66
FoReaL-Decoding	n=∞, p=0.2	30.0	12241	0.204	4.84	75.0	6502	0.216	2.43
FoReaL-Decoding	n=∞, p=0.4	46.7	11906	0.417	7.37	85.0	6719	0.423	4.07
FoReaL-Decoding	n=∞, p=0.6	50.0	11515	0.605	9.69	92.5	5671	0.607	4.42
FoReaL-Decoding	n=∞, p=0.8	60.0	10538	0.798	10.83	92.5	5925	0.797	5.87
FoReaL-Decoding	n=∞, p=1.0	66.7	13035	1.000	15.72	95.0	6244	1.000	7.54
DeepSeek-R1-Distill-Qwen-1.5B + Qwen2.5-7B-Instruct									
DeepSeek-R1-Distill-Qwen-1.5B		23.3	18021	-	2.86	65.0	13311	-	2.51
Qwen2.5-7B-Instruct		6.7	1243	-	0.95	52.5	994	-	0.75
FoReaL-Decoding	n=5, p=0.2	10.0	1047	0.170	0.73	50.0	923	0.179	0.64
FoReaL-Decoding	n=5, p=0.4	10.0	1381	0.230	0.91	55.0	1065	0.244	0.69
FoReaL-Decoding	n=5, p=0.6	13.3	2377	0.306	1.61	62.5	2574	0.302	1.97
FoReaL-Decoding	n=5, p=0.8	13.3	4203	0.345	2.87	47.5	2897	0.373	1.83
FoReaL-Decoding	n=5, p=1.0	16.7	7236	0.382	4.45	50.0	5614	0.428	3.24
FoReaL-Decoding	n=15, p=0.2	3.3	1936	0.208	1.38	47.5	985	0.224	0.65
FoReaL-Decoding	n=15, p=0.4	16.7	1189	0.339	0.70	45.0	1055	0.360	0.61
FoReaL-Decoding	n=15, p=0.6	16.7	1793	0.455	0.92	52.5	1307	0.482	0.65
FoReaL-Decoding	n=15, p=0.8	16.7	4120	0.545	2.05	57.5	2515	0.580	1.08
FoReaL-Decoding	n=15, p=1.0	16.7	14132	0.651	6.47	52.5	7330	0.686	3.35
FoReaL-Decoding	n=25, p=0.2	13.3	1243	0.249	0.80	50.0	958	0.231	0.62
FoReaL-Decoding	n=25, p=0.4	20.0	1317	0.389	0.73	42.5	1077	0.405	0.59
FoReaL-Decoding	n=25, p=0.6	16.7	1743	0.536	0.79	57.5	2047	0.560	1.14
FoReaL-Decoding	n=25, p=0.8	20.0	4474	0.693	1.57	65.0	3778	0.683	1.76
FoReaL-Decoding	n=25, p=1.0	23.3	11436	0.841	3.18	65.0	5721	0.865	1.66
FoReaL-Decoding	n=∞, p=0.2	13.3	1072	0.260	0.69	50.0	986	0.290	0.61
FoReaL-Decoding	n=∞, p=0.4	6.7	1276	0.420	0.68	42.5	1140	0.467	0.56
FoReaL-Decoding	n=∞, p=0.6	10.0	1914	0.614	0.78	57.5	1324	0.618	0.53
FoReaL-Decoding	n=∞, p=0.8	23.3	4244	0.788	1.26	65.0	2854	0.817	0.79
FoReaL-Decoding	n=∞, p=1.0	23.3	18021	1.000	2.86	65.0	13311	1.000	2.51