Making Small Language Models Efficient Reasoners: Intervention, Supervision, Reinforcement

Xuechen Zhang [*1] **Zijian Huang** [*1] **Chenshun Ni** [1] **Ziyang Xiong** [1] **Jiasi Chen** [1] **Samet Oymak** [1]

Abstract

Recent research enhances language model reasoning by scaling test-time compute via longer chain-of-thought traces. This often improves accuracy but also introduces redundancy and high computational cost, especially for small language models distilled with supervised fine-tuning (SFT). In this work, we propose new algorithms to improve token-efficient reasoning with small-scale models by effectively trading off accuracy and computation. We first show that the post-SFT model fails to determine the optimal stopping point of the reasoning process, resulting in verbose and repetitive outputs. Verbosity also significantly varies across wrong vs correct responses. To address these issues, we propose two solutions: (1) Temperature scaling (TS) to control the stopping point for the *thinking phase* and thereby trace length, and (2) TLDR: a length-regularized reinforcement learning method based on GRPO that facilitates multi-level trace length control (e.g. short, medium, long reasoning). Experiments on four reasoning benchmarks demonstrate that TS is highly effective compared to s1's budget forcing approach and TLDR significantly improves token efficiency by about 50% with minimal to no accuracy loss over the SFT baseline. Moreover, TLDR also facilitates flexible control over the response length, offering a practical and effective solution for token-efficient reasoning in small models.

1. Introduction

Recent works, such as OpenAI's o1, DeepSeek-R1, and s1 (Jaech et al., 2024; Guo et al., 2025; Muennighoff et al., 2025) have focused on enhancing the reasoning capabilities of language models by generating longer traces equipped with a *thinking* phase. These traces can be extremely long for challenging queries such as AIME or IOI problems. If we wish to make reasoning more efficient, we could either ask for a shorter reasoning trace or use a smaller language model to solve the problem at hand. This raises the question: Given a potentially small language model (SLM) and a maximum trace length, how can we achieve effective reasoning? In this work, we provide new insights and algorithms toward addressing this question. For training a small model, a go to approach is distillation where we use supervised fine-tuning (SFT) on the samples curated by a more powerful reasoning model like o1 or R1 (Guo et al., 2025). Recent works (Muennighoff et al., 2025; Guo et al., 2025; Li et al., 2025) indicate that this straightforward distillation approach on properly curated data significantly enhances the reasoning abilities of small models. We find that pure SFT has a key limitation: these models tend to generate excessively long answers that often contain repetitions. This highlights a major weakness of SFT compared to direct reward optimization via reinforcement learning (e.g. GRPO).

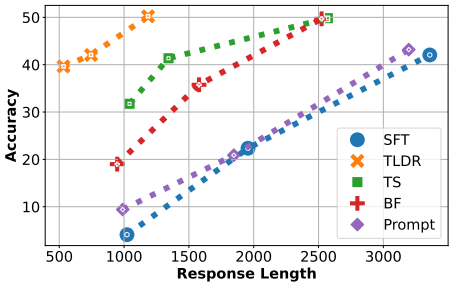


Figure 1. The average performances across four reasoning benchmarks for 7B models. Our RL-based length-control method TLDR enhances token efficiency by over 50% compared to SFT. Our temperature scaling (TS) method outperforms other test-time intervention techniques that avoid the need for training, such as Budget Forcing (BF) (Muennighoff et al., 2025) and Prompting. The details of baselines and our methods are provided in Sec. 3.

We advocate that optimizing the efficiency-accuracy tradeoff necessitates explicitly incorporating length regularization within reward or better intervention strategies if an RL training phase is not allowed. To this aim, we introduce *Trace Length Control for Dynamic Reasoning* (TLDR): a

[*]Equal contribution [1]EECS Department, University of Michigan, Ann Arbor, USA. Correspondence to: Xuechen Zhang <zxuechen@umich.edu>, Zijian Huang <zijianh@umich.edu>, Samet Oymak <oymak@umich.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

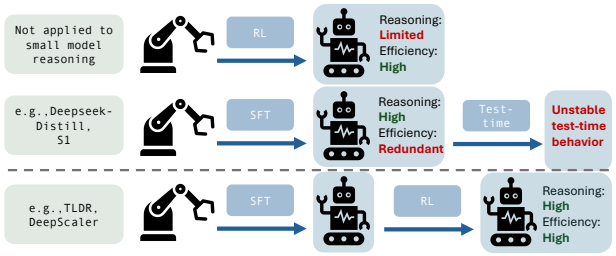


Figure 2. We explore training strategies for small language models (SLM), focusing on the effects on reasoning capability and test-time compute efficiency. RL improves efficiency, but when used alone, can compromise reasoning performance. While SFT enhances reasoning, it often leads to redundant outputs due to the lack of deliberate stopping point control. Test-time strategies offer limited control over output length and fail to improve performance with increased length consistently. To address these challenges, we propose TLDR, a method that combines SFT with RL to achieve both strong reasoning and token-efficient generation.

length-penalized variation of GRPO. Our evaluations show that, by utilizing a mild penalization, TLDR substantially reduces the average response length over GRPO while maintaining or even improving the overall accuracy. This could be viewed as a *free lunch* for the efficiency-performance trade-off. Additionally, we introduce a multi-level variation of TLDR that allows for dynamically controlling the response length by directly prompting the model. This eliminates the need for additional model training and offers a flexible approach to controlled generation. Experiments on multiple reasoning benchmarks demonstrate that our method significantly improves token efficiency and efficiency-performance pareto-front. As a training-free intervention method, we also introduce an intuitive temperature scaling strategy that provide a fine-grained control on the sampling of `end` token. Our specific contributions are as follows:

- **Rethinking SFT for distillation:** Through extensive experiments comparing SFT, SFT with test-time compute, and reinforcement learning (RL) across models of different sizes, we find that SFT-based methods result in substantial inefficiencies. These arise from redundant generation and repetitions, and notably can result in non-monotonic accuracy progress as the test-time compute budget increases (cf. Figure 8).
- **Temperature scaling to control stopping time in SFT models:** We propose TS as a training-free model-agnostic method to improve token-efficiency by controlling the sampling probability of the EOS token. Compared to budget forcing or textual prompting, this provides a finer-grained control and yields a better accuracy-efficiency pareto-front with up to 50% reduction in response length while maintaining accuracy across benchmarks.
- **Controlled generation via RL with multi-level length penalties** We introduce TLDR which integrates length-penalty within GRPO formulation to improve response efficiency. Remarkably, TLDR can reduce the response length while preserving overall performance compared to base GRPO. Within TLDR, we incorporate multiple penalty levels that can be specified by the user prompt. Through this, we show that TLDR can sweep the efficiency-accuracy pareto-front. Comparing pareto-fronts of different model sizes reveal scenarios where an SLM with longer trace (smaller penalty) outperforms an LLM with shorter trace while using fewer total FLOPs.

We remark that recent approaches (Muennighoff et al., 2025; Aggarwal & Welleck, 2025; Butcher et al., 2024; Yuan et al., 2024; Xu et al., 2025; Hou et al., 2025; Yang et al., 2025) have also highlighted the importance of length control. However, with the exception of the recent work L1 (Aggarwal & Welleck, 2025), these methods do not provide an optimized control over the average or maximum output length. Our evaluations show that lack of explicit optimization results in sub-optimal tradeoffs. TLDR compares on par with L1 under an max trace length constraint, while training from a weaker base model. We also contend that enforcing a maximum length constraint is often more desirable in real-world settings and refer the reader to Section 4 for details. The remainder of this paper is organized as follows. In Appendix B, we present our experimental characterization of SFT and RL strategies regarding their response length. Motivated by these observations, in Section 2 we present our training-free TS method and, in Section 3, we present our length penalty framework. Section 4 showcases the empirical results, and Section 5 concludes the paper with a discussion.

2. Proposed Method: Post-hoc Temperature Scaling

Motivated by the bad example shown in Figure 7, we check the next-token probabilities at the end of each block loop and find that in 87.5% of the cases, the end-of-sequence (EOS) token ranks among the top 5 candidates, despite not being selected. So, we introduce post-hoc temperature scaling to increase the probability of the end-of-sequence (EOS) token, to reduce the response length of the SFT-distilled models. The temperature scaling does not change the internal reasoning process of the model. Temperature scaling is a fundamental method for controlling model behavior, influencing aspects such as stochasticity of generative LLMs, calibration and imbalanced data, as highlighted in several studies (Zhang et al., 2024a; Menon et al., 2021; Li et al., 2021; Zhang et al., 2024c). $\mathbf{l} \in \mathbb{R}^V$ is the original logits output by the model, where V is the vocabulary size, and the EOS token has index i_{eos}, we modify the EOS logit as $l_{i_{\text{eos}}}{}' = \frac{z_{i_{\text{eos}}}}{T}, \quad T < 1$. By increasing the likelihood of EOS,

the model is more likely to terminate its generation earlier, thus producing shorter outputs. Empirically, we find that this simple adjustment effectively reduces redundancy, especially repeating, while preserving the core reasoning steps and answer accuracy. The result is shown in Table 1.

	Base		BF		TS	
	acc	length	acc	length	acc	length
SFT-S1-7b	77.00	4842.68	77.17	3591.21	77.01	1983.93
SFT-DeepSeek-1.5b	80.60	5869.60	81.03	3162.49	81.09	2615.66
SFT-DeepSeek-7b	88.17	4078.26	88.03	2839.95	88.95	2547.42

Table 1. Results of post-hoc temperature scaling. Since budget forcing (BF) requires a predefined target length, we sweep across 500, 1k, 2k, and 4k tokens, and select the config that achieves better performance than the base model while producing the shortest output as the BF baseline.

Based on the above observations, we ask two research questions:

- **QS1:** Given a small reasoning model, can we obtain a sweet spot between accuracy and response length, which means a model generates shorter responses without hurting the reasoning ability?
- **QS2:** Given a small reasoning model, can we incorporate the model with the ability to generate responses in different length levels by following the users' prompts?

3. Proposed Method: Reinforcement Learning with Length Penalty

To tackle the above two research questions, we propose penalized reward function, conditioned on the response length. The proposed reward functions can integrate with the Group Relative Policy Optimization (GRPO) algorithm, a popular RL algorithm for efficient training nowadays. We denote the response length penalty function as $\zeta(L)$, where L is the length of the response. The final reward function is $r = \hat{r} - \zeta(L)$, where $\hat{r}$ is the original reward function (e.g., accuracy reward, format reward, etc.).

A unique aspect of our approach is that different parameter settings can be indirectly controlled by the end user through a special prompt, for example "[Response Length: LONG] Provide a detailed step-by-step solution." In other words, the model is trained with the prompt and its corresponding penalty function. The model thus learns to pair the special prompt with the trajectories associated with that length penalty. Then during inference, the model should automatically produce responses that match that length penalty.

"Sweet Spot" of Length Penalty. To find the "sweet spot" of response efficiency without hurting the reasoning performance, we design the penalty function as $\eta(L) = \alpha \frac{L}{L_{\max}}$, where α is a hyperparameter for the strength of the length penalty, and $L_{\max}$ is the maximum length of the model's response. The α value should be set to a relatively small value to ensure that the penalty term $\eta(L)$ does not perturb the final reward r too much. Note that in this setting, the reward function $r = \hat{r} - \eta(L)$ correspondingly.

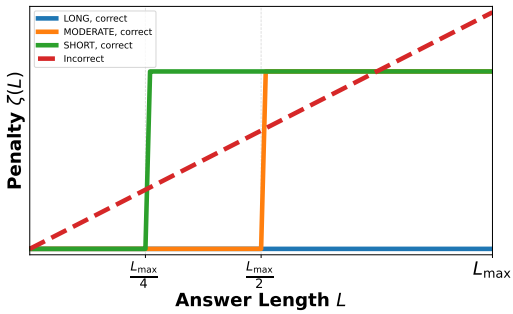


Figure 3. Length Penalty function.

Length Penalty for Multi-level Length Control. To address the problem of multi-level LP control, we define three different LP levels: short, moderate and long. Intuitively, the penalty function should be designed to penalize correct answers from being too long. We also set a special penalty ηL for wrong answers, which is same as "Sweet Spot" experiments, since we don't want LLMs to waste tokens when it cannot solve the problem with the current model size. Formally, the length penalty is defined as:

$$\zeta(L) = \begin{cases} 0, & \text{if } c = 1 \text{ and (LONG or (MODERATE and } L \leq \frac{L_{\max}}{2}\text{)} \\ & \text{or (SHORT and } L \leq \frac{L_{\max}}{4}\text{)}, \\ \beta, & \text{if } c = 1 \text{ and (MODERATE and } L > \frac{L_{\max}}{2} \\ & \text{or (SHORT and } L > \frac{L_{\max}}{4}\text{)}, \\ \eta(L), & \text{if } c = 0 \end{cases} \tag{1}$$

where LONG, MODERATE, and SHORT correspond to the desired LP level (paired with the corresponding prompts), $L > \frac{L_{\max}}{2}$ and $L > \frac{L_{\max}}{4}$ are the max length threshold corresponds to MODERATE, and SHORT prompts, $c \in \{0, 1\}$ means the answer is incorrect/correct, and β represents the penalty for correct answers whose length is greater than a threshold. The function are shown in Figure 3.

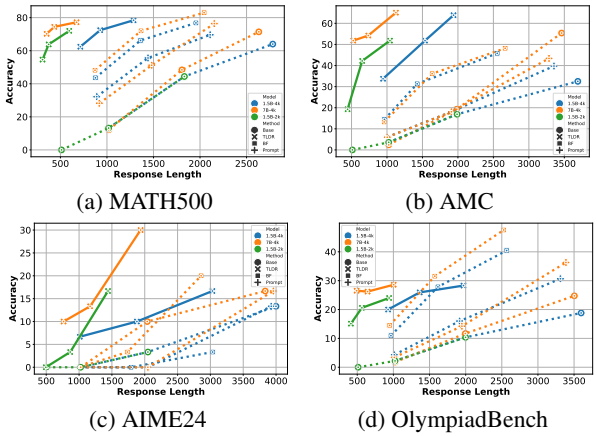
4. Experiments

4.1. Sweet Spot through Reinforcement Learning + Length Penalty

In Table 2, we report the accuracy and token length for the base model and for different settings of the length penalty. We can see in the second column "RL" that RL training directly within 4096 response length limitation can shorten the response length directly, while at the same time generally improving or maitaining the accuracy of SFT models compared to the Base model. From the third "RL + Length Penalty" column, we can see that by applying length penalty, we can further shorten the response length while preserving the accuracy. With the comparison between the results of

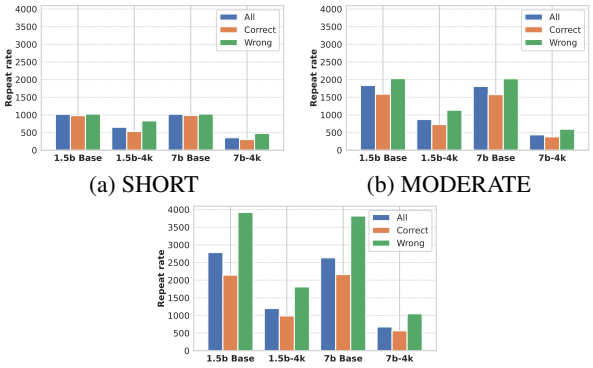
	Base		RL		RL+Length Penalty	
	acc	length	acc	length	acc	length
Qwen2.5-Math-1.5B(4k)	23.4	1976.874	68.2	685.896	71.4	495.842
Qwen2.5-Math-7B(4k)	57.2	1109.13	70	671.58	71.2	404.962
DeepSeek-R1-Distill-Qwen-1.5B(32768)	80.6	5769.596	77.2	1929.208	80.4	1104.748

Table 2. We investigate a range of models (size, short/long CoT, SFT/RL) and show that different values of the length penalty can produce shorter responses while maintaining good accuracy.



(a) MATH500 (b) AMC (c) AIME24 (d) OlympiadBench

Figure 4. Performance of TLDR under varying model sizes and token budget. TLDR outperforms length-controlled baselines (S1) both in performance and token efficiency.



(a) SHORT (b) MODERATE (c) LONG

Figure 5. The response length of models before and after trained with TLDR in different length control levels (SHORT/MODERATE/HIGH)). After training TLDR, both of the correct and wrong responses' length decreases.

"RL" and "RL + Length Penalty", we can see that while RL without length penalty can already gain better response efficiency, a better "Sweet Spot" can be obtained when training with length penalty.

4.2. Length Control through Reinforcement Learning + Multi-Level Length Penalty

Figure 4 shows the accuracy and token length for different models and settings of the length levels. From the results, we can see that the base DeepSeek-R1-Distill models tend to struggle with these strong reasoning tasks. This issue can be more severe when the response length is limited, which is shown by the accuracy gap between Figures 4a and 4c for the base model, and the results reported in DeepSeek-R1 (Guo et al., 2025). From Figure 4, we can see that after training the distilled models by GRPO with TLDR, we not only achieve much better accuracy on all four reasoning datasets by at most 50%, but shorten the response length by at most 50%. In particular, the lowest length penalty (LONG) generally results in the best accuracy albeit with slightly more tokens. We beat the base model, prompt control method and even the SOTA s1 (Muennighoff et al., 2025) both in accuracy and inference efficiency. In Figure 5, we show that both correct response length and wrong response length are shortened after training with TLDR, and the shortened ratio grows with the response length limitation. This means that TLDR can incorporate small language models both the ability to reduce redundancy during reasoning but also stop early when the problem is too hard for it. The detailed experiment results are at Table 4.

5. Conclusion

In conclusion, we propose a novel approach to improving token efficiency in reasoning tasks, addressing the inefficiencies of long reasoning traces generated by language models. Our work demonstrates that supervised fine-tuning (SFT) alone is insufficient to avoid redundant or repetitive outputs, especially for smaller models. To overcome this, we introduce TLDR, a length-penalized variation of Group Relative Policy Optimization (GRPO), which effectively reduces response length while maintaining accuracy. The flexibility of TLDR, through its multi-level penalty system, allows for dynamic control of response length, making it adaptable to different reasoning scenarios. Our experiments across four reasoning benchmarks show that TLDR achieves a significant improvement in token efficiency—around 50%—without a noticeable loss in accuracy. Furthermore, TLDR's performance compares favorably to existing methods, offering a superior trade-off between efficiency and accuracy. This work provides valuable insights into optimizing the computational cost of reasoning models, and the proposed method is a practical step toward more efficient and scalable reasoning systems.

Acknowledgements

This work is supported by the National Science Foundation grants CCF-2046816, CCF-2403075, CCF-2212426, the Office of Naval Research grant N000142412289, and an Adobe Data Science Research Award. The computational aspects of the research is generously supported by computational resources provided by the Amazon Research Award on Foundation Model Development.

References

Aggarwal, P. and Welleck, S. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Akyürek, E., Damani, M., Zweiger, A., Qiu, L., Guo, H., Pari, J., Kim, Y., and Andreas, J. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Butcher, B., O'Keefe, M., and Titchener, J. Precise length control in large language models. *arXiv preprint arXiv:2412.11937*, 2024.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Google. Gemini 2.0 flash thinking mode (gemini-2.0f lash-thinking-exp-1219), 2024. https://cloud.google.com/vertex-ai/generative-ai/docs/thinking.

Gozeten, H. A., Ildiz, M. E., Zhang, X., Soltanolkotabi, M., Mondelli, M., and Oymak, S. Test-time training provably improves transformers as in-context learners. *arXiv preprint arXiv:2503.11842*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gupta, N., Narasimhan, H., Jitkrittum, W., Rawat, A. S., Menon, A. K., and Kumar, S. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*, 2024.

Hou, B., Zhang, Y., Ji, J., Liu, Y., Qian, K., Andreas, J., and Chang, S. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, M., Zhang, X., Thrampoulidis, C., Chen, J., and Oymak, S. Autobalance: Optimized loss functions for imbalanced data. *Advances in Neural Information Processing Systems*, 34:3163–3177, 2021.

Li, Y., Yue, X., Xu, Z., Jiang, F., Niu, L., Lin, B. Y., Ramasubramanian, B., and Poovendran, R. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*, 2025.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W. Y., Roongta, M., Cai, C., Luo, J., Zhang, T., Li, L. E., Popa, R. A., and Stoica, I. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

Menon, A. K., Jayasumana, S., Rawat, A. S., Jain, H., Veit, A., and Kumar, S. Long-tail learning via logit adjustment. *ICLR*, 2021.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/~.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Setlur, A., Rajaraman, N., Levine, S., and Kumar, A. Scaling test-time compute without verification or rl is suboptimal. *arXiv preprint arXiv:2502.12118*, 2025.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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

Xu, Y., Dong, H., Wang, L., Sahoo, D., Li, J., and Xiong, C. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

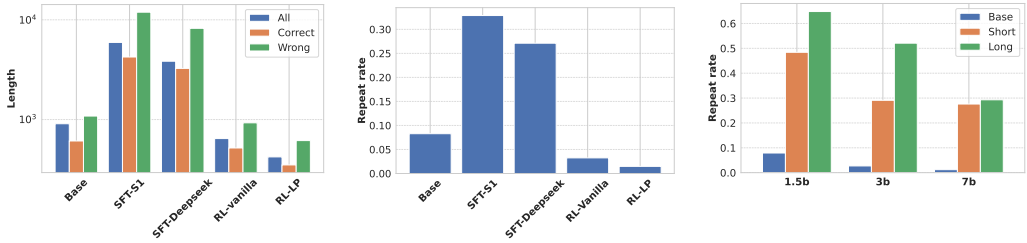
Yang, W., Ma, S., Lin, Y., and Wei, F. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*, 2025.

Yuan, W., Kulikov, I., Yu, P., Cho, K., Sukhbaatar, S., Weston, J., and Xu, J. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*, 2024.

Zhang, X., Chang, X., Li, M., Roy-Chowdhury, A., Chen, J., and Oymak, S. Selective attention: Enhancing transformer through principled context control. *Advances in Neural Information Processing Systems*, 37:11061–11086, 2024a.

Zhang, X., Huang, Z., Taga, E. O., Joe-Wong, C., Oymak, S., and Chen, J. Efficient contextual LLM cascades through budget-constrained policy learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=aDQlAz09dS.

Zhang, X., Li, M., Chen, J., Thrampoulidis, C., and Oymak, S. Class-attribute priors: adapting optimization to heterogeneity and fairness objective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16890–16898, 2024c.



(a) Response length for different FT strategies

(b) Repeat rate for different FT strategies

(c) Repeat rate for different model sizes and FT trace length

Figure 6. SFT models have longer and more repetitive answers when wrong, especially for small models fine-tuned under supervision by long traces.

A. Related Work

Efficient LLMs. LLMs are widly popular for tasks such as conversational AI, code generation, healthcare, and scientific research. However, their increasing size and computational demands pose challenges for scalability, efficiency, and deployment. This has become a central challenge with the rising importance of test-time compute scaling (Snell et al., 2024) where the model generates long chain-of-thought traces (Muennighoff et al., 2025; Jaech et al., 2024; Guo et al., 2025) or conducts other computation/training (Akyürek et al., 2024; Gozeten et al., 2025) to solve challenging problems. Speculative decoding (Leviathan et al., 2023) accelerates autoregressive generation by leveraging draft models to predict multiple tokens in parallel. MoE-based architectures (Shazeer et al., 2017) reduces computational cost while maintaining performance by activating only a subset of parameters per input. Quantization techniques, such as GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024), compress models into lower precision formats, achieving speed-ups with minimal accuracy loss. Cascade-based methods (Chen et al., 2023; Zhang et al., 2024b; Gupta et al., 2024) save the inference cost by choosing the appropriate models and prompts by either rule-based selection or training an RL algorithm. Instead of complicated model architecture designs or multi model dependence, we build a simple method to make language models more efficient by training from common models.

RL vs SFT. To obtain smaller language models with relatively high performance, Deepseek-R1-distill family (Guo et al., 2025) shows that smaller language models can also get high performance by SFT with the long CoT reasoning answer generated by the strong reasoning LLM. s1 (Muennighoff et al., 2025) shows that SFT on only 1,000 examples suffices to build a competitive reasoning model matching o1-preview and produces a model that lies on the pareto frontier. However, (Setlur et al., 2025) argues that scaling test-time compute without verification or RL is suboptimal, meaning that distillation samll models do not understand the reasoning process but replicate the solution. We demonstrate that only SFT can either make the model generate meaningless and redundant responses when querying difficult questions or do not reach the best trade-off point between reasoning ability and efficiency.

Length Control in LLMs. To fulfill different hardware settings and different users' requirements, several recent studies focus on controlling the reasoning or response length, to have the ability either to save the inference cost by limiting the length or to dig out the reasoning potential of language models. s1 (Muennighoff et al., 2025) utilizes a inference strategy called budget forcing on the distilled small model to control the thinking length, while L1 (Aggarwal & Welleck, 2025) empower the model with length control by adding penalty term with different token length targets during RL training on a super powerful 1.5b model. There are also a few contemporaneous works (Xu et al., 2025; Hou et al., 2025; Yang et al., 2025). Compared to these, we focus on and contrast the impact of different strategies (intervention, SFT, RL) on reasoning length control. As a result, we provide temperature scaling as a more natural and effective intervention strategy compared to Budget Forcing. Compared to L1 which uses DeepScaler as their base model, we provide a stronger understanding of the shortcomings of SFT and demonstrate that RL can unlock token-efficiency even with weaker base models.

B. Observations of Supervised Finetuning and Reinforcement Learning Regarding Response Length

In this section, we scrutinize a range of models, including Qwen-2.5 family (Yang et al., 2024a;b), Deepseek-R1-distill family (Guo et al., 2025), DeepScaleR (Luo et al., 2025) and our RL fine-tuned models. The objective of this section is to

Okay, so I need to find the domain of the function log(x2). The problem states that the domain is x < a or x > b, and I have to find a + b. Hmm, let me think. ..... and the domain is x < a or x > b.

But the problem is asking for a + b. So, if a is less than 0 and b is greater than 0, then a + b would be some real number. However, the problem doesn't specify any particular values for a or b.
Wait, but let me think again. The problem says ""the domain of the function log x2 is x < a or x > b"". So, the domain is all real numbers except x=0. Therefore, a and b must be such that a < 0 and b > 0, and the domain is x < a or x > b.

Figure 7. Example of wrong and lengthy response, generated by the Deepseek-R1-Distill-Qwen-1.5B model for a Math500 question. The response repeatedly cycles through the paragraph shown in the blue block until it reaches the maximum context length, without providing a final answer.

explore the benefits and drawbacks of the two post-training finetuning methods, SFT and RL, in terms of their reasoning capability and efficiency. We make four main observations below, which later inform the application of our intervention method in Section 2 and the design of our TLDR framework in Section 3.

B.1. Observation 1: Supervised fine-tuning introduces redundancy

To equip more efficient smaller models with reasoning capabilities, open-source models, like Qwen (Yang et al., 2024a;b), with SFT, use samples curated by powerful RL-trained long reasoning models such as Deepseek-R1 (Guo et al., 2025). Recent works indicate that this straightforward distillation approach significantly enhances the reasoning abilities of small models. Our main observation is that LLMs with SFT sometimes generate excessively long answers, especially for smaller models.

Experiment setup. We use the Qwen-2.5 family (Yang et al., 2024a;b) as the base models as well as short CoT models. For SFT models, we use the Deepseek-R1-distill family (Guo et al., 2025), labeled as "SFT-Deepseek", which has been fine-tuned on the Deepseek-R1 curated dataset containing approximately 800k samples of reasoning and non-reasoning data. Additionally, we study the S1 family of models (Muennighoff et al., 2025), labeled "SFT-S1", which has been fine-tuned on S1K, the dataset containing 1k questions paired with reasoning traces. To obtain S1 SFT models of different sizes, we use the officially released S1K SFT models. For models fine-tuned with RL, we train using either standard GRPO or our proposed (described later in Section 3). We evaluate on the MATH500 dataset (Lightman et al., 2023).

SFT models generate excessively long and repetitive responses. First, we investigate the two SFT models with improved performance compared to the Qwen base model. The results of models with 7b parameters are shown in Table 3. We also visualize the length distribution in Figure 6a. In Figure 6a, the blue bars are the average length of all the responses, the orange bars are the average length of all the correct responses and the green bars show the average length of all the incorrect responses. We can see that SFT models generate longer responses compared to the base model and RL fine-tuned models. Further, the average length of wrong answers are always longer than correct answers. Such responses are undesirable because they are costly, inefficient, and unlikely to be correct. An example wrong answer is shown in Figure 7.

To understand the prevalence of such repetitive answers, we compute the "repeat rate", defined as the proportion of incorrect answers that are excessively long and repetitive among all wrong answers. Repetition is identified by querying GPT-4o-mini (OpenAI, 2024) with a custom prompt (Appendix F). The repeat rate results are presented in Figure 6b. We can see that the repeated error rate of SFT models is much higher than that of base model and RL fine-tuned models. We also propose hypothesis why RL natually provides more effective control in Appendix B.5.

B.2. Observation 2: Longer traces for FT causes redundant responses for small models

Experiment setup. We include the S1 family of models (Muennighoff et al., 2025), which has been fine-tuned on S1K or S1K-1.1, two datasets containing 1k questions paired with reasoning traces. These traces were generated using either the Google Gemini Flash Thinking API (Google, 2024) (S1K) or Deepseek-R1 (S1K-1.1), which are labeled as "short" and "long" respectively in Figure 6c. Notably, reasoning traces generated by Deepseek-R1 are of higher quality but are also significantly longer, often including the "aha" moment that encourages deeper reasoning. The length distribution of the two kinds of traces are shown in Appendix H.

Longer traces for FT hurt response efficiency. Previous research (Muennighoff et al., 2025) suggests that incorporating

difficulty, and high-quality traces can enhance reasoning performance. However, we argue that complex traces can decrease the efficiency of small models and may even adversely affect performance, as they might exceed the maximum context length before completing their reasoning process. In Figure 6c, we present the repeat rate for different model sizes and trace lengths. The orange bars indicate the results of models trained with shorter Gemini traces, while the green bars show models trained with significantly longer Deepseek traces. We observe that smaller models tend to exhibit higher repeat rates. Additionally, using long reasoning traces for fine-tuning increases this rate. Among all incorrect responses, over 60% of failures are due to repetition in the 1.5B parameter model that was fine-tuned with supervision using long Deepseek traces. We provide an example of a repeating incorrect answer in Appendix I, Figure 15. In this example, the model generates the word "wait" 577 times, continuously repeats the same sentences, and fails to provide a final answer before reaching the maximum context length, with one paragraph repeating 180 times. Even worse, when the model finally escapes this repetitive cycle, it often falls back into it. We argue that fine-tuning with high-quality, longer traces severely hurts response efficiency.

B.3. Observation 3: Controlling context length trades off performance for efficiency

Experiment setup. We experiment with three different sizes of state-of-the-art SFT models, drawn from the Deepseek-R1-distill models (Guo et al., 2025) on the MATH500 dataset (Lightman et al., 2023) with a state-of-the-art S1 "budget forcing" method (Muennighoff et al., 2025) that caps the thinking trajectory length using test-time compute. "Auto" denotes responses generated without any extra constraints on response length.

The results are shown in Figure 8a for different model sizes. The x-axis denotes the average context length, and the y-axis shows the corresponding accuracy. Dots on the upper left of the figure have a better efficiency-performance trade-off. "Budget forcing" achieves shorter response lengths with comparable or even better performance compared to responses generated without constraints ("Auto") across all three sizes. The improvement of budget forcing methods is most significant on the smaller model, showing an almost 50% length decrease on Deepseek-R1-distill-1.5b (blue). However, without training, such test-time length control strategies sometimes fail to find an optimal trade-off (results shown in the Appendix in Figure 12). These results suggest significant room for improvement in achieving an efficient performance trade-off, a key motivation for our TLDR.

B.4. Observation 4: Test-time compute cannot precisely control response length

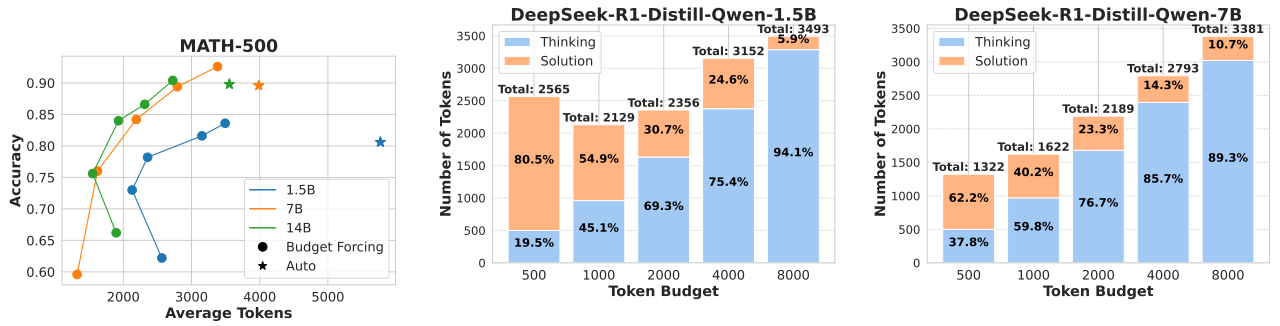
Experiment setup. We conducted experiments using various test-time length control strategies, including the S1 "budget forcing" as in the previous subsection, and "exact control". These methods force the thinking trajectory to be of fixed length through a collection of strategies: suppressing the generation of the end-of-thinking token delimiter to enforce a minimum response length, appending a "Wait" to the model's current reasoning trace to encourage reflection on its generation (Aggarwal & Welleck, 2025)), and adding the prompt "Think for up to n tokens" after the question. These experiments were performed on three different datasets: MATH500 (Lightman et al., 2023), GPQA (Rein et al., 2024), and AIME24.

The results on the MATH500 dataset using budget forcing are shown in Figure 8. The budget forcing method cannot precisely control the total response length, as demonstrated in Figure 8a. Despite a maximum 500-token constraint for the thinking trajectory, the final response often extends to about 2000 tokens or more (first green/blue dot), which is longer than the responses generated with a maximum 1000-token constraint. This is because forcing a shorter thinking trajectory sometimes leads to a significantly longer final solution. To illustrate this, we plot the ratio of tokens spent during thinking vs the final solution in Figures 8b and 8c, for different token budgets. These length control methods performed poorly. Additional results for all length control strategies and the datasets are in Figures 12 and 13 in the Appendix.

B.5. Observation 5: SFT introduces redundancy due to equal treatment of end token, while RL-based models can learn when to stop

In SFT, the model is trained to predict the next token based on ground truth sequences, with each token, including special tokens such as end-of-sequence (EOS), <think>, <answer>, treated equally during optimization. This method can lead to redundancy in reasoning tasks for several reasons:

- **Lack of Explicit Stopping Signal:** Since the EOS token is treated like other tokens, the model is not explicitly encouraged to optimize for brevity. Instead, it learns to imitate long reasoning traces, even when shorter reasoning suffices.



(a) Efficiency-performance trade-off

(b) Thinking:solution token ratio of Deepseek-R1-distill-1.5b

(c) Thinking:solution token ratio of Deepseek-R1-distill-7B

Figure 8. Test time compute strategies can trade off performance for efficiency, but cannot precisely control response length, in part due to many tokens spent on the final solution.

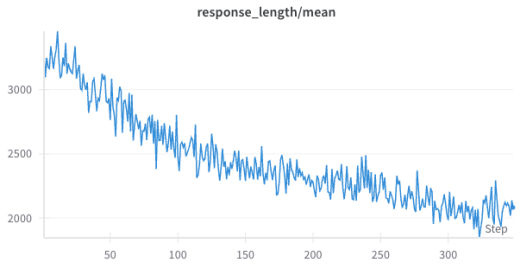


Figure 9. Changes in response length during reinforcement training without length penalty.

- **Bias:** During training, the model only sees EOS tokens once at the end of the reasoning traces.

- **Overfitting to Long Traces:** If the training data contains long, detailed reasoning sequences, SFT models tend to memorize these structures instead of learning when a concise answer would be sufficient.

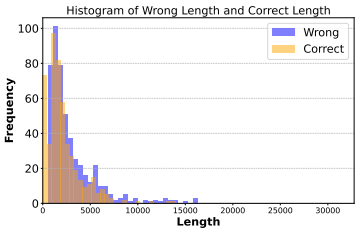
RL can provide more effective control. Unlike SFT, RL-based models can learn when to stop reasoning. RL treats stopping as a decision-making problem instead of treating it equally with other tokens. The model can learn when to terminate reasoning to maximize rewards. This is evidenced by decreases in response length during training, as shown in Figure 9.

In RL, we can also define a reward function that explicitly balances reasoning quality and efficiency. Unlike SFT, where the model passively learns from long training sequences, RL allows us to actively penalize excessive length, while rewarding correct and concise answers, by adding a length penalty.

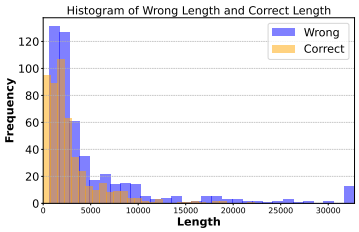
C. Experiment Setup

C.1. Setup for Sweet Spot Experiments

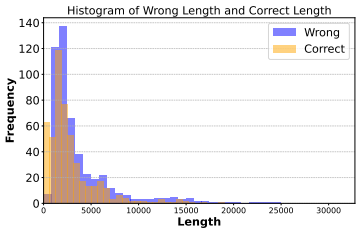
In this experiment, we use 3 base models as shown in Table 2, which contains models of different types and sizes (1.5b and 7b). Specifically, Qwen2.5-Math-1.5B, Qwen2.5-Math-7B (Yang et al., 2024a;b) are short CoT reasoning models, while DeepSeek-R1-Distill-Qwen-7B is a long CoT reasoning model. We warm up the models by training them on the MATH training dataset with GRPO, where $\hat{r} = 0.9 \cdot r_{\text{acc}} + 0.1 \cdot r_{\text{format}}$, r_{acc}, r_{format} are the reward for accuracy and format, and then train with length penalty as defined in **"Sweet Spot" of Length Penalty**, where $\alpha \in \{0.0, 0.1\}$ for RL without length penalty and RL with length penalty correspondingly. After training finishes, we test the models' performance on the MATH500 dataset. For all RL training, we set the response max length to 4096 because this is max length that can be trained with our resource, and 4096 is generally long enough to solve problems in MATH dataset.



(a) DeepScaleR-1.5B



(b) DeepSeek-R1-Distill-1.5B



(c) DeepSeek-R1-Distill-7B

Figure 10. The length distribution of wrong and correct responses of different models. The length of correct responses are generally shorter than the length of wrong responses.

C.2. Setup for TLDR Experiments

In these experiments, we use two base models: DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B. The training dataset is a combination of the training set of MATH, AIME, AMC, STILL, OlympiadBench, which is same as the training set of DeepScaleR-1.5B-Preview. We first warm up the models by training them to the convergence in the sense of the original reward $\hat{r}$, then we do the training with multi-level response length penalty as defined in Equation (1), where $\gamma = 0.1, \beta = 0.3$ in our setting. After training finishes, we test the models' performance on the MATH500 dataset and the test sets of AMC, AIME24 and OlympiadBench datasets. For DeepSeek-R1-Distill-Qwen-1.5B, we experiment with $L_{\text{max}} = 2048, 4096$, and for DeepSeek-R1-Distill-Qwen-7B, we experiment with $L_{\text{max}} = 4096$. The prompts for low/medium/high length penalty are "[Response Length: LONG] Provide a detailed step-by-step solution.", "[Response Length: MODERATE] Provide a concise but clear solution.", "[Response Length: SHORT] Provide only the essential steps." respectively. These three levels corresponds to the $\frac{1}{4}, \frac{1}{2}$ length and the full length of the max response length.

D. Model performance

Table 3 shows the performance of different models on three reasoning datasets, where Qwen2.5-{}b-Instruct_s1K models are Qwen2.5-{}b-Instruct supervised finetuned with s1K dataset, DeepSeek-R1-Distill-Qwen-{}B models are distilled models officially released by DeepSeek, s1.1-3B models are models released by the s1 team. From Table 3, we can see that distillation with a small amount of high quality curated long CoT data can have comparable performance when the model size is larger (7B and 14B) and the task is not very difficult (MATH and GPQA), but it's still not close to the distilled models with a large amount of long when the model is small (1.5B, 3B, 7B) or the task is reasoning intensive (AIME).

Model	MATH Accuracy	GPQA Accuracy	AIME Accuracy
Qwen2.5-1.5b-Instruct_s1K	0.426000	0.176768	0.066667
Qwen2.5-3b-Instruct_s1K	0.579158	0.247475	0.066667
Qwen2.5-7b-Instruct_s1K	0.740000	0.166667	0.100000
Qwen2.5-14b-Instruct_s1K	0.794000	0.414141	0.233333
Qwen2.5-32b-Instruct_s1K	0.870000	0.601010	0.366667
DeepSeek-R1-Distill-Qwen-1.5B	0.818000	0.404040	0.400000
DeepSeek-R1-Distill-Qwen-7B	0.881764	0.515152	0.366667
DeepSeek-R1-Distill-Llama-8B	0.856000	0.510101	0.533333
DeepSeek-R1-Distill-Qwen-14B	0.903808	0.558376	0.586207
DeepSeek-R1-Distill-Qwen-32B	0.914000	0.595960	0.833333
s1.1-3B	0.630000	0.287879	0.133333
s1.1-7B	0.770000	0.393939	0.266667
s1.1-14B	0.840000	0.575758	0.300000
s1.1-32B	0.896000	0.621212	0.433333

Table 3. Model Performance

E. Model length distribution

Figure 10 illustrates the token length distributions for correct and incorrect responses on the MATH-500 dataset across three different models: (a) DeepScaleR-1.5B, (b) DeepSeek-R1-Distill-1.5B, and (c) DeepSeek-R1-Distill-7B. Across all models, correct responses tend to have shorter lengths on average compared to incorrect ones.

```
prompt = (
    f"Please analyze the following text and determine if there are any
    meaningless repetitions of identical sentences. "
    f"Note that just phrases with 'wait' or 'Wait' itself is not considered a
    repetition."
    f"Only consider it a repetition if the same content appears much more than
    10 times. "
    f"Provide 'yes' only if you find exact repetitions, and in the next line, please
    specify the repeated sentence and specify its count in a new line.
    Otherwise, return 'no'. Text: {answer}"
)
```

Figure 11. The prompt used for detecting answer repetition with GPT-4o-mini.

F. Repeat Prompt

Figure 11 shows the prompt that we use to identify repetition in Figure 6b when using GPT-4o-mini.

G. Test time strategy can not preciously control context length

We consider four test-time strategies for controlling the length of the thinking trajectory:

1. **Budget Forcing (BF):** A maximum token budget is imposed on the thinking trajectory. Within this limit, the model is free to decide when to generate a special token indicating the end of thinking and the start of the final answer. If the model fails to do so before reaching the budget, the end-of-thinking token is forcibly inserted, and the model proceeds to generate the final answer.

2. **Exact Control (EC):** The thinking trajectory is forced to be of a fixed length. If the model prematurely generates the end-of-thinking token before reaching the desired length, the token is removed, and generation continues until the target length is reached. At that point, the end-of-thinking token is forcibly appended to trigger final answer generation.

3. **Prompt Control (PC):** A soft constraint is applied by including an instruction in the prompt that explicitly tells the model not to exceed a specified number of tokens for the thinking trajectory.

4. **Auto:** The model is left unrestricted, allowing it to autonomously decide when to terminate the thinking trajectory and begin generating the final answer.

In Figure 12, we can see that none of the test time strategies can exactly control the response under the length limitation. What makes things worse is that the accuracy improvement rate is much slower than the response length growth rate when encountering harder reasoning tasks such as AIME and GPQA.

H. Analysis of S1K trace

Figure 14 shows the distribution of response lengths of Gemini and DeepSeek on the s1K dataset. We can see that DeepSeek generates longer reasoning traces when compared with Gemini. This makes its responses more suitbale for reasoning distillation to obtain a higher accuracy. while also introduce more severe redundancy for small language models at the same time.

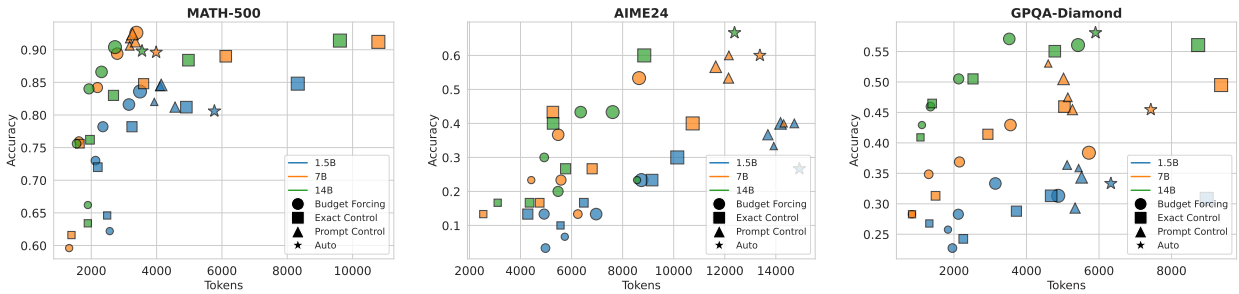
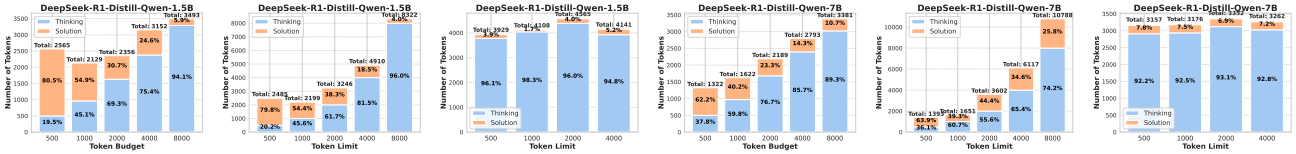


Figure 12. Each subplot shows the performance of different models (1.5B, 7B, and 14B) on three benchmarks: MATH-500, AIME24, and GPQA-Diamond. The x-axis represents the average number of tokens in the generated response (including both thinking trace and final answer), while the y-axis indicates accuracy. Within each shape category, marker size reflects variations of the same strategy (e.g., Budget Forcing-500 to Budget Forcing-8000), with larger markers indicating higher token budgets.

I. Bad example

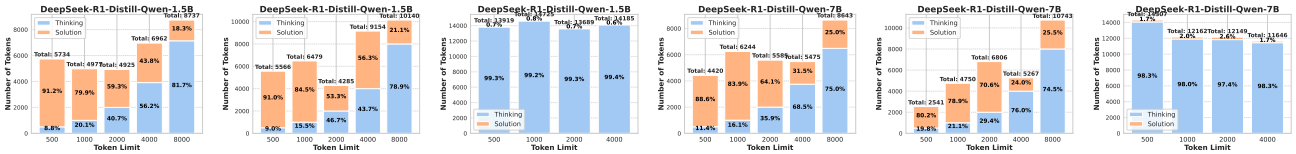
Figure 15 shows a typical failure case where the model exhibits severe repetition issue when generating reasoning for AIME24 Question 4.

J. Additioned Experiment Results



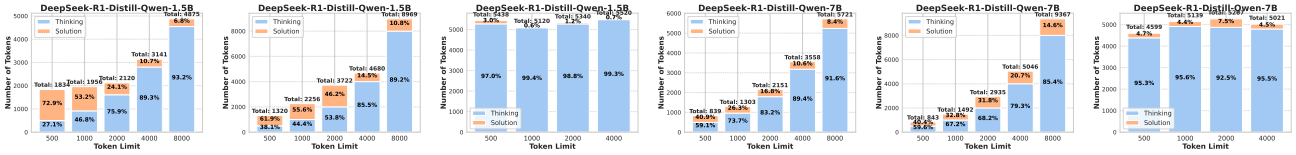
(a) 1.5B-BF (b) 1.5B-EC (c) 1.5B-PC (d) 7B-BF (e) 7B-EC (f) 7B-PC

Figure 13. MATH-500



1.5B-BF (h) 1.5B-EC (i) 1.5B-PC (j) 7B-BF (k) 7B-EC (l) 7B-PC

Figure 13. AIME24



1.5B-BF (n) 1.5B-EC (o) 1.5B-PC (p) 7B-BF (q) 7B-EC (r) 7B-PC

Figure 13. GPQA-Diamond

Figure 13. Length distribution of thinking/solution trajectories under different test-time strategies (Budget Forcing, Exact Control, Prompt Control), model sizes (1.5B vs 7B), and benchmarks (AIME24, MATH-500, GPQA-Diamond).

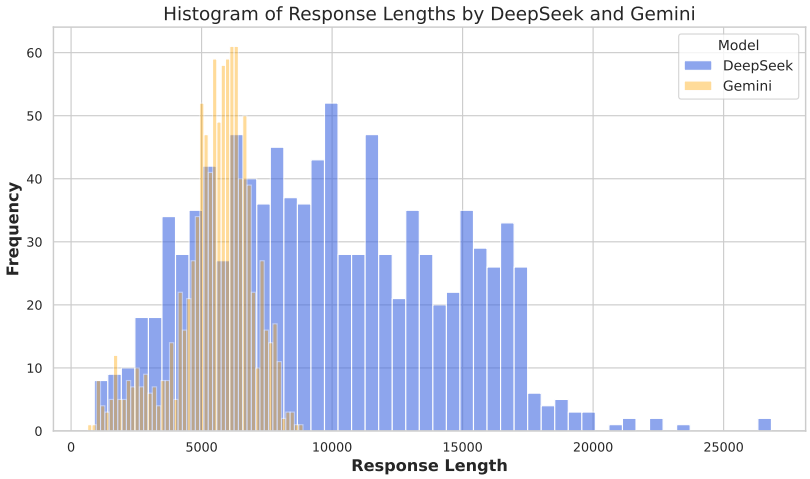


Figure 14. Histograms of response lengths by Gemini and DeepSeek on s1K dataset.

<|im_start|>think\nOkay, so Jen enters a lottery by picking 4 distinct numbers from the set S = {1, 2, 3, ..., 9, 10}. Then 4 numbers are randomly chosen from S. She wins a prize if at least two of her numbers are among the randomly chosen ones, and the grand prize if all four are hers. We need to find the probability that Jen won the grand prize given that she won a prize. The answer should be in the form m/n where m and n are coprime, and then find m + n.\n\nFirst, let me parse the problem. The total number of possible 4-number combinations from S is C(10,4). For each combination, 4 numbers are drawn randomly.

.....

Wait, but actually, if a combination has two numbers in D, then it's included in the subsets of D that contain those two numbers. However, each combination can be written as E ∪ F where E is a subset of D and F is a subset of D.

Wait, actually, the number of combinations with at least two numbers in D is equal to the number of subsets of D with size 2, 3, or 4. Each such subset is counted once, because each combination that has two numbers in D is included in exactly one subset of D with size 2, three numbers in D is included in exactly one subset of size 3, and four numbers in D is included in exactly one subset of size 4. Wait, but actually, if a combination has two numbers in D, then it's included in the subsets of D that contain those two numbers. However, each combination can be written as E ∪ F where E is a subset of D and F is a subset of D.

repeat 180 times

<|im_start|>answer\nFinal Answer: To solve the problem of finding the probability that Jen won the grand prize given that she won a prize, we need to compute the conditional probability \\(\\frac{\\text{Number of combinations with at least two numbers matching the drawn set}}{\\text{Total number of combinations}}\\).\n\nFirst, we note that the total number of possible combinations is \\(C(10, 4) = 210\\). The grand prize is winning all four numbers, which occurs with probability \\(\\frac{1}{210}\\).\n\nTo find the number of combinations with at least two numbers matching the drawn set, we use the principle of inclusion-exclusion.

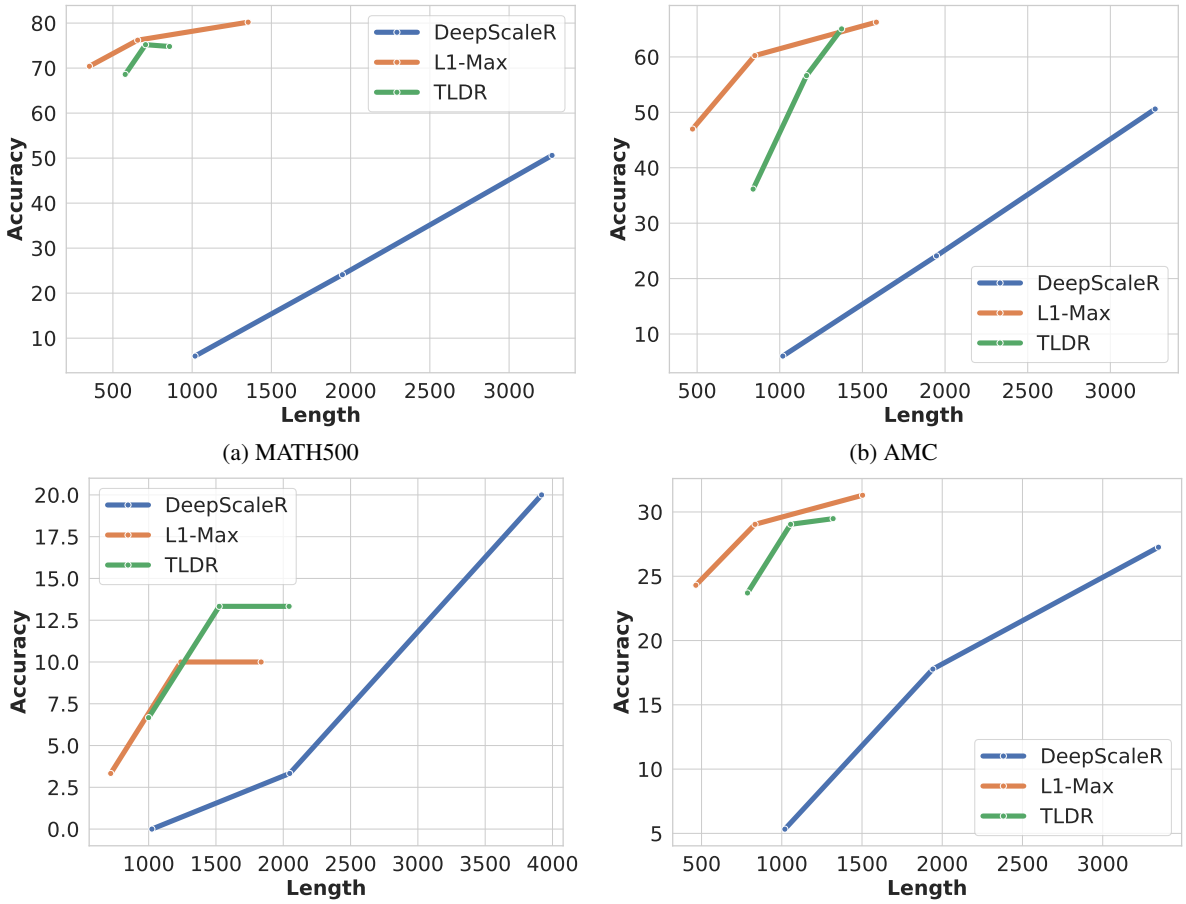
.....

The conditional probability is \\(\\frac{\\frac{1}{210}}{\\frac{23}{42}} = \\frac{1}{210} \\times \\frac{42}{23} = \\frac{42}{210 \\times 23} = \\frac{2}{21 \\times 23} = \\frac{2}{483}\\).

But the problem states that the number of combinations with at least two numbers matching is 115, which includes the grand prize. Therefore, the conditional probability is \\(\\frac{1}{23}\\). However, the problem states that the number of combinations with at least two numbers matching is 115, which includes the grand prize. Therefore, the conditional probability is \\(\\frac{1}{23}\\).\n\n

repeat 34 times

Figure 15. The response generated by Qwen-Instruct-1.5b finetuned by S1K-1.1 on AIME24 problem 4.



(a) MATH500

(b) AMC

(c) AIME24

(d) OlympiadBench

Figure 16. TLDR compares on par with L1 under an max trace length constraint. Moreover, TLDR requires only a single training phase of 100 epochs, whereas L1 involves a more complex two-phase training process: 700 epochs for the first phase (L1-Exact) and an additional 120 epochs for the second phase (L1-Max).

	No Length Control						With Length Control					
	SHORT		MODERATE		LONG		SHORT		MODERATE		LONG	
	acc	length	acc	length	acc	length	acc	length	acc	length	acc	length
DeepSeek-R1-Distill-Qwen-1.5B-2k	0.0	512	13.2	1017.756	44.4	1832.292	54.6	307.862	63.8	373.63	72	595.232
DeepSeek-R1-Distill-Qwen-1.5B-4k	13.2	1017.756	44.4	1832.292	64.0	2781.848	60.0	650.08	65.0	868.134	74.4	1194.83
DeepSeek-R1-Distill-Qwen-1.5B-4k-new							62.4	714.17	72.4	927.476	78.4	1285.882
DeepSeek-R1-Distill-Qwen-7B-4k	12.2	1019.408	48.4	1805.768	71.4	2631.856	70.2	351.828	74.4	433.952	77.2	673.16
DeepScaleR-1.5B-Preview-4k	28.8	966.116	59.4	1618.526	74	2329.458	68.6	577.702	75.2	706.082	74.8	857.298
L1-Qwen-1.5B-Max-4k	NA						70.4	351.774	76.2	656.242	80.2	1353.576
S1-Qwen-1.5B-BudgetForcing-4k	NA						43.60	876.31	66.20	1360.79	76.80	1936.26
S1-Qwen-7B-BudgetForcing-4k	NA						48.20	871.75	72.00	1366.49	83.00	2044.31
DeepSeek-R1-Distill-Qwen-1.5B-prompt-4k	NA						32.20	890.22	55.60	1441.21	69.60	2108.98
DeepSeek-R1-Distill-Qwen-7B-prompt-4k	NA						28.20	918.25	51.20	1480.86	76.40	2153.89

(a) MATH500

	No Length Control						With Length Control					
	SHORT		MODERATE		LONG		SHORT		MODERATE		LONG	
	acc	length	acc	length	acc	length	acc	length	acc	length	acc	length
DeepSeek-R1-Distill-Qwen-1.5B-2k	0.0	512	3.6	1024	16.9	1988.036	19.28	444.47	42.17	650.19	51.81	1041.90
DeepSeek-R1-Distill-Qwen-1.5B-4k	3.6	1024	16.9	1988.036	32.5	3690.096	32.53	939.39	46.48	1423.66	59	2043.96
DeepSeek-R1-Distill-Qwen-1.5B-4k-new							33.73	944.53	51.81	1536.16	63.86	1937.98
DeepSeek-R1-Distill-Qwen-7B-4k	2.4	1024	19.3	1984.229	55.4	3461.916	51.81	524.42	54.22	736.25	65.06	1125.60
DeepScaleR-1.5B-Preview-4k	6.02	1018.48	24.10	1948.16	50.60	3271.06	36.15	838.92	56.63	1162.20	65.06	1374.27
L1-Qwen-1.5B-Max-4k	NA						46.99	472.37	60.24	848.96	66.27	1584.44
S1-Qwen-1.5B-BudgetForcing-4k	NA						14.46	964.06	31.33	1624.02	45.78	2553.99
S1-Qwen-7B-BudgetForcing-4k	NA						13.25	960.19	36.14	1636.60	48.19	2665.64
DeepSeek-R1-Distill-Qwen-1.5B-prompt-4k	NA						6.02	999.94	18.07	1906.54	39.76	3354.06
DeepSeek-R1-Distill-Qwen-7B-prompt-4k	NA						6.02	1003.98	18.07	1908.93	43.37	3284.07

(b) AMC

	No Length Control						With Length Control					
	SHORT		MODERATE		LONG		SHORT		MODERATE		LONG	
	acc	length	acc	length	acc	length	acc	length	acc	length	acc	length
DeepSeek-R1-Distill-Qwen-1.5B-2k	0	512	0	1024	3.33	2048	0	494.4	3.33	865.8	16.67	1441.37
DeepSeek-R1-Distill-Qwen-1.5B-4k	0	1024	3.33	2048	13.33	3999.83	6.67	1007.83	13.3	1702.57	10	2637.7
DeepSeek-R1-Distill-Qwen-1.5B-4k-new							6.67	1011.17	10	1880.8	16.67	3030.8
DeepSeek-R1-Distill-Qwen-7B-4k	0	1024	10	2038.13	16.67	3835.0	10	764.33	13.33	1168.2	30	1936.93
DeepScaleR-1.5B-Preview-4k	0	1024	3.33	2048	20	3919.03	6.67	999.53	13.33	1524.2	13.33	2043.4
L1-Qwen-1.5B-Max-4k	NA						3.33	718.2	10	1238.8	10	1835.83
S1-Qwen-1.5B-BudgetForcing-4k	NA						0	1004.10	0	1792.73	3.33	3033.50
S1-Qwen-7B-BudgetForcing-4k	NA						0	1007.87	3.33	1737.03	20	2857.97
DeepSeek-R1-Distill-Qwen-1.5B-prompt-4k	NA						0	1024.00	0	2048.00	13.33	3926.93
DeepSeek-R1-Distill-Qwen-7B-prompt-4k	NA						0	1024.00	0	2048.00	16.67	3959.57

(c) AIME24

	No Length Control						With Length Control					
	SHORT		MODERATE		LONG		SHORT		MODERATE		LONG	
	acc	length	acc	length	acc	length	acc	length	acc	length	acc	length
DeepSeek-R1-Distill-Qwen-1.5B-2k	0	512	2.2	1023.4	10.37	1999.238	15.11	412.58	20.59	564.63	24	941.90
DeepSeek-R1-Distill-Qwen-1.5B-4k	2.2	1023.4	10.37	1999.238	18.81	3598.139	16.88	850.82	20.30	1245.40	25.48	1951.65
DeepSeek-R1-Distill-Qwen-1.5B-4k-new							20	926.65	25.93	1363.68	28.30	1969.88
DeepSeek-R1-Distill-Qwen-7B-4k	1.78	1023.25	11.7	1996.39	24.74	3504.83	26.52	491.54	26.22	643.62	28.59	1004.45
DeepScaleR-1.5B-Preview-4k	5.33	1019.20	17.78	1941.47	27.26	3347.61	23.70	786.27	29.04	1054.94	29.48	1319.93
L1-Qwen-1.5B-Max-4k	NA						24.30	464.51	29.04	833.43	31.3	1503.4
S1-Qwen-1.5B-BudgetForcing-4k	NA						10.96	965.88	27.85	1616.35	40.44	2567.32
S1-Qwen-7B-BudgetForcing-4k	NA						14.52	948.51	31.56	1573.08	47.56	2526.31
DeepSeek-R1-Distill-Qwen-1.5B-prompt-4k	NA						4.30	1012.55	16.00	1915.82	30.67	3311.35
DeepSeek-R1-Distill-Qwen-7B-prompt-4k	NA						3.41	1011.88	14.22	1948.05	36.30	3387.5311

(d) OlympiadBench

Table 4. Detailed performance of different models and different methods in four reasoning datasets. The first 3 rows are the results of original DeepSeek distilled models (No Length Control) and models finetuned with TLDR, L1-Qwen-1.5B-Max-4k is the method in (Aggarwal & Welleck, 2025), 2 rows with S1 is the method proposed in (Muennighoff et al., 2025), and the last 2 rows are controlling original DeepSeek distilled model by prompts. In the name of models, "2k" and "4k" means the max length of model output, and "SHORT", "MODERATE" and "LONG" means the $\frac{1}{4}$, $\frac{1}{2}$ and the original max length. For different datasets, RL with our multi-level length penalty design can shorten response length while achieving good accuracy, particularly for the "long" length penalty setting.

	SHORT			MODERATE			LONG		
	All	Correct	Wrong	All	Correct	Wrong	All	Correct	Wrong
Deepseek+TLDR	714.17	611.71	884.21	927.476	772.36	1334.36	1285.882	1091.87	1990.06
DeepScaleR+TLDR	1374.28	1145.57	1800.14	1374.28	1145.57	1800.14	1374.28	1145.57	1800.14
DeepScaleR	966.116	831.90	1020.40	1618.526	1367.84	1985.29	2329.458	1857.26	3673.39
L1	351.774	291.85	494.30	656.242	568.94	935.76	1353.576	1287.46	1621.39
Deepseek	1017.756	978.788	1023.682	1832.292	1587.640	2027.662	2781.848	2142.125	3919.133

(a) MATH500

	SHORT			MODERATE			LONG		
	All	Correct	Wrong	All	Correct	Wrong	All	Correct	Wrong
Deepseek+TLDR	944.53	842.43	996.51	1536.16	1229.0	1866.35	1937.98	1607.68	2521.5
DeepScaleR+TLDR	838.92	585.37	982.43	1162.20	919.89	1478.56	1374.28	1145.57	1800.14
DeepScaleR	1018.48	932.4	1024.0	1948.16	1633.65	2048.0	3271.06	2494.64	4066.41
L1	472.37	417.26	521.23	848.96	736.02	1020.09	1584.45	1477.13	1795.25
Deepseek	1024.0	1024.0	1024.0	1988.036	1692.5	2048.0	3690.096	2848.222	4096.0

(b) AMC

	SHORT			MODERATE			LONG		
	All	Correct	Wrong	All	Correct	Wrong	All	Correct	Wrong
Deepseek+TLDR	1011.17	1021.5	1010.43	1880.8	1677.0	1903.44	3030.8	2228.0	3191.36
DeepScaleR+TLDR	999.53	878.5	1008.18	1524.2	1343.75	1551.96	2043.4	1437.5	2136.62
DeepScaleR	1024.0	NA	1024.0	2048.0	2048.0	2048.0	3919.03	3686.0	3977.29
L1	718.2	772.0	716.34	1238.8	1117.0	1252.33	1835.83	1551.0	1867.48
Deepseek	1024.0	NA	1024.0	2048.0	2048.0	2048.0	3999.83	3396.25	4092.69

(c) AIME24

	SHORT			MODERATE			LONG		
	All	Correct	Wrong	All	Correct	Wrong	All	Correct	Wrong
Deepseek+TLDR	926.64	793.81	959.85	1363.68	1038.28	1477.568	1969.88	1394.24	2197.04
DeepScaleR+TLDR	786.27	559.76	856.64	1054.94	750.94	1179.33	1319.93	848.86	1516.87
DeepScaleR	1019.20	957.11	1022.70	1941.47	1632.28	2008.33	3347.61	2397.49	3703.67
L1	464.51	362.54	497.24	833.43	656.24	905.92	1503.47	1329.72	1581.93
Deepseek	1023.4	1003.66	1023.87	1999.238	1684.94	2035.60	3598.139	2574.03	3835.478

(d) OlympiadBench

Table 5. Average response length of models' responses in 4 datasets. We report the value in different levels and in the whole dataset, subset of correct responses and subset of wrong responses. TLDR can reduce the response by at most 50% from base models.

	SHORT	MODERATE	LONG
Deepseek+TLDR	0.0053	0.0507	0.1296
DeepScaleR+TLDR	0.0127	0.0725	0.0635
DeepScaleR	0.0168	0.0936	0.2692
L1	0.0135	0.0168	0.1313
Deepseek	0.0138	0.1798	0.4111

(a) MATH500

	SHORT	MODERATE	LONG
Deepseek+TLDR	0.0000	0.0750	0.1333
DeepScaleR+TLDR	0.0189	0.1389	0.1379
DeepScaleR	0.0128	0.0635	0.3170
L1	0.0227	0.0606	0.0714
Deepseek	0.0000	0.2029	0.5000

(b) AMC

	SHORT	MODERATE	LONG
Deepseek+TLDR	0.0357	0.0370	0.2000
DeepScaleR+TLDR	0.0357	0.1923	0.2308
DeepScaleR	0.0000	0.1379	0.1667
L1	0.0345	0.0742	0.2592
Deepseek	0.0333	0.2759	0.3462

(c) AIME24

	SHORT	MODERATE	LONG
Deepseek+TLDR	0.0333	0.0720	0.1756
DeepScaleR+TLDR	0.0330	0.1023	0.1492
DeepScaleR	0.0125	0.1333	0.2770
L1	0.0117	0.0355	0.1290
Deepseek	0.0167	0.1983	0.3923

(d) OlympiadBench

Table 6. Repeat rate of models' responses in 4 datasets. Repeat rate is calculated using GPT-4o to identify significant repetitions of identical sentences in each response, excluding short phrases like 'wait'. TLDR can reduce the response by at most $\frac{2}{3}$ from base models.

	SHORT	MODERATE	LONG
Deepseek+TLDR	3.516	4.114	5.73
DeepScaleR+TLDR	3	3.806	4.16
DeepScaleR	4.856	7.768	11.062
L1	2.848	4.272	7.826
Deepseek	4.22	7.72	11.64

(a) MATH500

	SHORT	MODERATE	LONG
Deepseek+TLDR	4.75	6.75	8.81
DeepScaleR+TLDR	4.25	5.89	7.12
DeepScaleR	5.82	10.13	16.45
L1	3.86	5.76	9.57
Deepseek	4.60	8.57	15.73

(b) AMC

	SHORT	MODERATE	LONG
Deepseek+TLDR	5.27	7.7	13.67
DeepScaleR+TLDR	6.03	9.97	10.23
DeepScaleR	6.6	11.53	20.13
L1	6.57	10.73	13.07
Deepseek	5.27	9.37	18.97

(c) AIME24

	SHORT	MODERATE	LONG
Deepseek+TLDR	4.74	6.45	9.11
DeepScaleR+TLDR	4.31	5.51	6.55
DeepScaleR	5.69	10.45	17.43
L1	3.67	6.08	9.57
Deepseek	4.39	8.94	16.93

(d) OlympiadBench

Table 7. Steps of models' responses in 4 datasets. We define the number of steps as the number of occurrence of keywords in the keyword list ["But", "Wait,", "Alternatively,", "Perhaps", "First,", "Okay,", "Given", "The", "Therefore,", "So,"]. TLDR can reduce the response by at most 50% from base models.