Latent Chain-of-Thought? Decoding the Depth-Recurrent Transformer

Wenquan Lu[1], **Yuechuan Yang**[1], **Kyle Lee**[1], **Yanshu Li**[1], **Enqi Liu**[2]
[1]Brown University, [2]Harvard University
wenquan_lu@brown.edu

Abstract

Chain-of-thought (CoT) reasoning has enabled transformer-based language models to excel at complex mathematics and multi-step planning. However, in standard decoder-only architectures, these reasoning steps are externalized in natural language, improving interpretability at the cost of efficiency. To capture reasoning that is not easily represented in words, many works have explored recurrent architectures that aim to internalize reasoning in latent space, potentially supporting latent CoT. In this paper, we investigate whether such reasoning structures emerge in Huginn-3.5B, a depth-recurrent Transformer that reuses layers at inference time without increasing parameter count. We examine the model's internal behavior on arithmetic tasks using a suite of probing techniques including the Logit Lens and Coda Lens. Our findings reveal limited evidence of interpretable latent CoT by tracking rank trajectories of final and intermediate result tokens. Furthermore, we uncover significant probing inconsistencies across recurrent blocks, where the interpretability of hidden states depends heavily on both the layer index and the decoding method. Finally, we empirically show that increasing recurrence depth yields only marginal gains and falls well short of models that explicitly externalize reasoning steps. The code is available at https://github.com/wenquanlu/huginn-latent-cot.

1 Introduction

Modern large language models demonstrate remarkable capabilities in reasoning and planning tasks (Guo et al., 2025; Yu et al., 2023). Much of this success relies on Chain-of-thought (Wei et al., 2022; Zhang et al., 2024; Chen et al., 2024): explicitly prompting the model to articulate intermediate steps in natural language. This strategy, though effective, may introduce verbosity and slow inference. A compelling alternative is to develop models that perform reasoning in latent space without surfacing intermediate steps in language. Yet, it remains unclear whether today's architectures are capable of such behavior.

A promising approach to latent reasoning leverages recurrent methods (Hao et al., 2024), where intermediate continuous hidden states are passed across reused layers to simulate multi-step reasoning without emitting language. The Huginn-3.5B model (Geiping et al., 2025) exemplifies this idea with a depth-recurrent Transformer that reuses layers at inference to increase computational depth per token. While increasing recurrences improves performance on reasoning tasks, it remains unclear whether this stems from iterative refinement or the emergence of structured, CoT-like reasoning in latent space (Yang et al., 2024).

As a result, in this paper, we ask: *Does Huginn exhibit signs of latent chain-of-thought reasoning during inference?* To investigate this, we conduct a systematic analysis of Huginn's hidden states on arithmetic tasks under conditions that suppress explicit reasoning. We introduce an unrolled view of the architecture and apply a range of probing techniques including logit lens, coda lens, and token rank trajectory tracking to decode and visualize the model's internal computations. In summary, our key contributions and findings are as follows.

1. **We uncover significant probing inconsistencies across blocks in Huginn's depth-recurrent architecture.** Unlike the smoothly evolving representations typically observed in

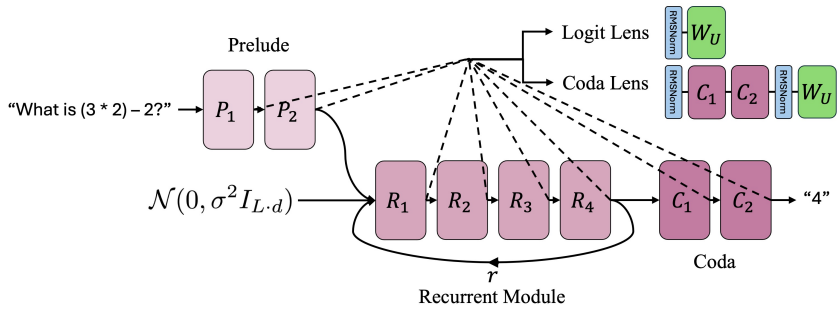

Figure 1: Overview of our approach to decode hidden states in the depth-recurrent Huginn. For each block's output, we employ logit lens and coda lens to convert them into logits.

feedforward Transformers (nostalgebraist, 2021), Huginn exhibits sharp discontinuities in hidden state semantics across layers. In particular, different blocks (e.g., R1 vs. R4) encode distinct information, and their interpretability strongly depends on the choice of decoding lens. 2. **Token rank trajectory analysis provides little evidence for latent CoT reasoning.** By tracing the rank dynamics of both intermediate and final answer tokens, we find no clear temporal separation or structured latent reasoning pathway across recurrence steps, contrary to what latent CoT would predict. 3. **Scaling recurrence depth fails to match explicit reasoning.** On the GSM8K benchmark, increasing the number of recurrent steps only marginally improves performance and falls far short of models that leverage explicit CoT prompting, highlighting the limitations of depth recurrence *alone* for inducing effective reasoning behavior.

2 Method

In this section, we first introduce an unrolled view of the Huginn architecture. We then present the two main approaches for probing and decoding the hidden states of the depth-recurrent transformer to detect latent chain-of-thought (CoT): logit lens and coda lens.

2.1 Unrolled View of Huginn Architecture

As shown in fig. 1, the architecture of Huginn 3.5B model consists of 2 Prelude blocks $\{P_1, P_2\}$, 4 Recurrent Blocks $\{R_1, R_2, R_3, R_4\}$ and 2 Coda blocks $\{C_1, C_2\}$, where each block is a standard, causal self-attention block. Given input tokens $x \in \mathbb{R}^{L \times |V|}$, the input is first embedded by the embedding matrix $W_E \in \mathbb{R}^{|V| \times d}$ to input embeddings $e = xW_E, e \in \mathbb{R}^{L \times d}$. These embeddings are first processed by the Prelude blocks, followed by r recurrent passes through the Recurrent blocks, and finally the Coda blocks for prediction. By unrolling the recurrence, the input embeddings are passed through $2 + 4r + 2$ blocks, where r denotes the number of recurrent steps taken during a single forward pass for next-token prediction. The hidden states s_i produced by each block can be summarized in the following equation:

$$s_i = \begin{cases} e & i = 0 \\ P_i(s_{i-1}) & 1 \leq i \leq 2 \\ R_1(s_2, n), \quad n \sim \mathcal{N}(0, \sigma^2 I_{L \cdot d}) & i = 3 \\ R_{(i-3) \bmod 4+1}(s_{i-1}) & 4 \leq i \leq 2+4r,\ i \not\equiv 3 \pmod 4 \\ R_1(s_2, s_{i-1}) & 4 \leq i \leq 2+4r,\ i \equiv 3 \pmod 4 \\ C_{i-(2+4r)}(s_{i-1}) & 2+4r+1 \leq i \leq 2+4r+2 \end{cases} \tag{1}$$

Note that in the third line of eq. (1), a random vector drawn from normal distribution is used as the initial state for the recurrence. From this unrolled perspective, there are $2 + 4r + 2$ hidden states which we can track the trajectory of intermediate computations.

2.2 Decoding Hidden States by Logit Lens and Coda Lens

Logit lens. The logit lens is a widely used technique for interpreting intermediate representations in transformer-based models. As illustrated in fig. 1, for each hidden state s_i, we first apply RMS normalization, followed by projection through the unembedding matrix $W_U \in \mathbb{R}^{d \times |V|}$ to obtain logits z_i in the vocabulary space. These steps align with Huginn's architectural choice of normalizing features prior to unembedding:

$$z_i = \text{RMSNorm}(s_i) W_U \tag{2}$$

We focus on the logits from the last token position $z_i[-1]$ which associates with model's current prediction. Prior work has shown that such projections often yield interpretable, top-ranked tokens that are aligned with the model's internal computation (Geva et al., 2023; 2022). Thus, analyzing $z_i[-1]$ across intermediate layers or recurrent steps provides insight into how the model's predictions evolve over time.

Coda Lens. The Huginn model's recurrent architecture includes a specialized coda module $C = \{C_1, C_2\}$ to effectively map the output of the last recurrent block to logits during inference. The coda block is a more expressive decoder compared to logit lens as it consists of two transformer blocks. Accordingly, we also decode the hidden states using this learned module, a method we refer to as the *coda lens*, to explore whether it produces more faithful or semantically aligned logits:

$$z_i = \text{RMSNorm}(C(\text{RMSNorm}(s_i)))W_U \tag{3}$$

In line with Huginn's implementation, we apply normalization both before and after the coda module to ensure consistency. As with the logit lens, we focus our analysis on $z_i[-1]$.

3 Experiment

3.1 Experimental Setup for Mathematical Reasoning Without Explicit CoT

Datasets. In our main experiment, we use the synthetic arithmetic test data employed in GPT-3 (Brown et al., 2020). Specifically, We use the **one-digit composite** task: the model is asked to perform a composite operation on three 1 digit numbers. For example, "Question: What is (9 + 8) * 2? Answer: 34". We use such a simple dataset for evaluation and analysis because the Huginn model only achieves an accuracy of 0.19 on the dataset. The dataset has in total 2k questions. To ensure negative results are not artifacts of errors, we later restrict our analysis to correctly answered subsets. In addition, we also test model's performance on the standard mathematical reasoning dataset GSM8K (Cobbe et al., 2021) which contains 8.5K high quality grade school math word problems.

Suppress Explicit CoT. To encourage latent chain-of-thought (CoT), in all experiments, we suppress explicit CoT by enforcing the model to output the answer straightway using system message and four in-context examples (see appendix A.2). We format system message as "You are a concise and helpful assistant. Always return only the final answer straightway." In in-context examples, the final answer of the question are given as correct output without any additional token. Unless otherwise mentioned, we set recurrent steps to 16 for all experiments. So in total there are $2 + 4 \times 16 + 2 = 68$ blocks.

3.2 Discontinuities in Hidden State Interpretability of Depth-Recurrent Transformer

As a first step in our investigation, we examine whether the outputs decoded by the logit lens and coda lens across Huginn's unrolled layers exhibit the pattern of **initial guess followed by smooth refinement** observed in conventional decoder-only language models (Vaswani et al., 2017), a phenomenon originally identified through logit lens analysis, where models quickly reach a coarse prediction in early layers, then progressively converge toward the final prediction (nostalgebraist, 2021). Since Huginn is a depth-recurrent transformer, it is unclear whether such interpretability patterns still hold. If this pattern were observed, it would provide strong evidence against the presence of latent CoT dynamics, as smooth refinement implies continuous convergence rather than discrete internal transitions. However, as we

show below, Huginn exhibits markedly different behavior. To test this, we select the first 100 arithmetic questions from the arithmetic dataset and run Huginn forward **once** on each question, recording the average rank of the final predicted tokens decoded from each unrolled block. The trajectories of ranks through unrolled layers are visualized in fig. 2.

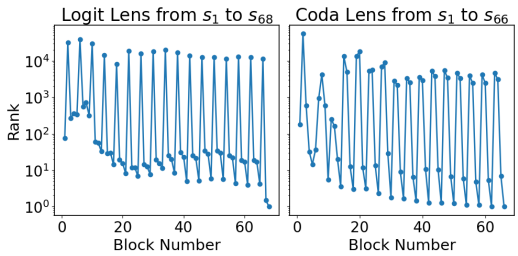

Figure 2: Average rank trajectory of the final predicted token (via logit lens (left), coda lens (right)) across unrolled blocks, averaged over 100 arithmetic questions. Note the first two ranks in each graph are from the prelude.

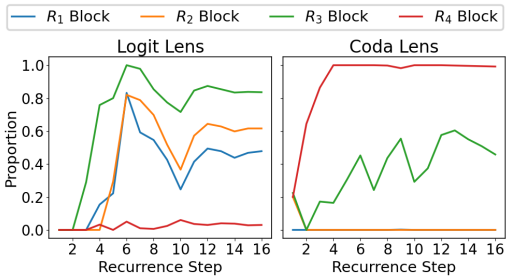

Figure 3: Proportion of top-5 decoded tokens (via logit lens (left), coda lens (right)) that are valid signed-integer prefixes across recurrence steps, averaged over 100 arithmetic questions.

As seen from fig. 2, the ranks exhibit large-magnitude, periodic oscillations as the token moves through layers. Note that we use logarithmic scale for vertical axis as the rank distributions vary substantially across layers. In fig. 2 (left), we observe consistent upward spikes at the R_4 layer, while most other layers exhibit low decoded ranks. This pattern reverses in fig. 2 (right), where the downward spikes consistently occur at R_4 layer, with high decoded ranks concentrated in the other layers.

The observation is further supported by fig. 3, where we measure the proportion of top-5 decoded tokens that are valid signed-integer prefixes (e.g., "5", "-", "2"). Because the 100 test questions are purely arithmetic, these prefixes are expected intermediate steps if the model engages in latent CoT reasoning. As shown in fig. 3 (left), a large proportion of top decoded tokens from blocks $\{R_1, R_2, R_3\}$ using the logit lens are signed numeric prefixes, while almost none from R_4 are. In contrast, for the coda lens fig. 3 (right), nearly 100% of tokens decoded from R_4 are signed numeric prefixes, while $\{R_1, R_2\}$ yield virtually none.

Manual inspection of top-5 tokens further illustrates the divergence both across blocks and between decoding methods. For example, R_4 decoded with logit lens produces uninterpretable outputs such as {`"inc"`, `"unity"`, `"friendships"`, `"igne"`, `"impulse"`}, whereas coda lens on exactly the same hidden state produces numerical tokens that closely relates to the arithmetic computation: {`"6"`, `"5"`, `"1"`, `"7"`, `"2"`}. **Conversely**, at the earlier block R_1, decoding with coda lens produces general-purpose outputs such as {`"answer"`, `"answers"`, `"tru"`, `"clarification"`, `"spa"`}. These tokens relate to the semantics of answering in general, rather than reflecting numerical computations. However, decoding via logit lens from exactly the same hidden state still results in a high proportion of numerical tokens: {`"5"`, `"3"`, `"1"`, `" answer"`, `"None"`}. Further examples are provided in appendix A.3.

These discoveries suggest that the interpretability of intermediate states in Huginn, when probed using logit and coda lens, varies dramatically depending on the block index and decoding method. There are clear discontinuities in representations across blocks, particularly around R_4. This shows top decoded tokens do not form a smooth convergence toward the final prediction over unrolled layers, and that **lens applicability must be assessed on a per-layer basis**. One possible explanation for the distinct behavior of R_4 is that its output can serve a dual role: feeding into both the next recurrent cycle via R_1 and the coda C_1, which may force it to encode a representation that differs markedly from other blocks.

3.3 Tracing Final and Intermediate Tokens Provides Little Evidence for Latent CoT

We now investigate whether Huginn exhibits latent CoT by tracing the rank trajectories of **signature tokens**: the intermediate and final result tokens in arithmetic problems. These

tokens serve as anchors for assessing whether the model performs multi-step reasoning across recurrence. In our setting of 1-digit composite task, e.g., 2 * 3 + 1, the intermediate result is 6, and the final result is 7. We filter the 2k arithmetic dataset to a subset of 67 questions using the following criteria (1) the model predicts correct answer. (2) the final result and any intermediate result are only single digit/token; the constraint improves the interpretability because the language model operates on token-level information. (3) the final result is different to intermediate result so their rank trajectory do not trivially overlap.

To address the blockwise inconsistency discovered in section 3.2 and prevent drastic oscillatons in graph, we restrict our analysis and visualization to R_3 outputs for logit lens, and R_4 outputs for coda lens. These blocks have best alignment with final prediction and can be decoded into interpretable domain as shown in section 3.2.

fig. 4 shows the rank trajectory of the final result token and intermediate result token across recurrent steps, we also include the rank of a random token 'the' as a baseline reference. If the model performs latent CoT, we expect the rank of the intermediate token to drop first, followed by a delayed drop in the final token's rank, reflecting stepwise reasoning. However, such phase separation is not observed in fig. 4. In both subplots, the ranks of both the final and intermediate tokens descend quickly in early recurrent steps, with the final token consistently maintaining a lower rank than the intermediate token. Hence, no clear evidence of latent CoT is observed based on the rank analysis. However, an interesting **rank reversal** between final and intermediate tokens at around step 6 is observed for most examples. This could potentially indicate the model is re-evaluating the final outcome based on intermediate results. We leave a deeper investigation of this phenomenon to future work. The graphs for other recurrent blocks are provided in appendix A.1, which also do not signal any latent CoT.

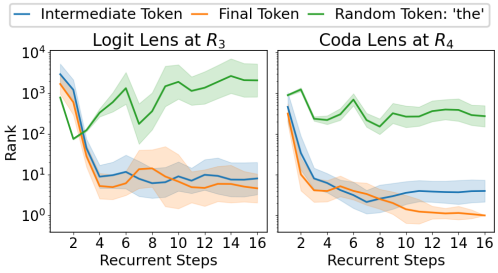

Figure 4: Rank trajectory of the final, intermediate, and random token (decoded via logit lens (left), coda lens (right)), averaged over 67 single-digit arithmetic questions that the model answers correctly. Shaded regions denote ±1 relative std.

Model	Recurrent Steps	GSM8K Accuracy
Huginn	64	**24.87/38.13**
Huginn w/o CoT	4	3.11/3.11
	8	4.47/4.47
	16	4.78/4.78
	32	4.93/4.93
	64	4.70/4.70
	128	4.93/4.93
	256	4.62/4.62

Table 1: GSM8K accuracy (strict/flexible) across different models and recurrence steps. Without explicit CoT, there is a monotonic increase in accuracy as recurrent steps increases from 4 to 32. However, it is still substantially lower than that with explicit CoT as shown in the first row of the table.

3.4 Scaling Recurrent Steps Cannot Beat Explicit Chain-of-Thought

Given the lack of clear evidence for latent chain-of-thought (CoT) reasoning in our earlier probing analysis, we turn to a macroscopic performance evaluation to detect any indirect traces of such behavior. Specifically, we benchmark Huginn on the GSM8K dataset. As in prior experiments, we suppress explicit CoT reasoning by modifying the system message. To remain consistent with the original Huginn evaluation setup, we use an 8-shot prompting format. As shown in table 1, increasing the number of recurrent steps from 4 to 32 leads to only modest gains in accuracy (from 3.11 to 4.93), and performance plateaus thereafter. In contrast, Huginn with explicit CoT achieves significantly higher accuracy (24.87/38.13). This suggests that even if some latent reasoning emerges within the recurrent loop, it is insufficient to rival the effectiveness of standard chain-of-thought reasoning.

4 Conclusion

In this paper, we investigated whether depth-recurrent transformers, exemplified by Huginn, exhibit latent chain-of-thought reasoning. We analyze the internal dynamics of the model on

arithmetic tasks under conditions that suppress explicit reasoning. Through logit lens and coda lens, we find little evidence of structured latent chain-of-thought reasoning. However, our results do not definitively rule out the presence of latent CoT. If it exists, it may be more subtle or distributed than our employed tools can detect. Future work may apply more advanced probing techniques, such as activation patching (Meng et al., 2022), to uncover finer-grained reasoning patterns potentially hidden within the recurrent loop.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shibo Hao, 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.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

nostalgebraist. Interpreting gpt: the logit lens, 2021. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 2025-03-21.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356, 2024.

A Appendix

A.1 Rank Trajectories of the Intermediate, Final and Random Tokens from Other Recurrent Blocks

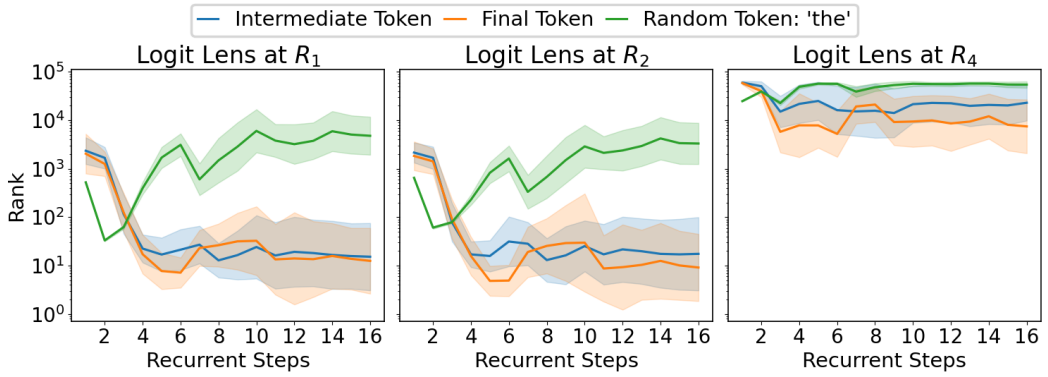

Figure 5: Rank trajectories of the intermediate, final and random tokens decoded by logit lens at recurrent blocks 1 (left), 2 (middle) and 4 (right). Shaded regions denote ± 1 relative std.

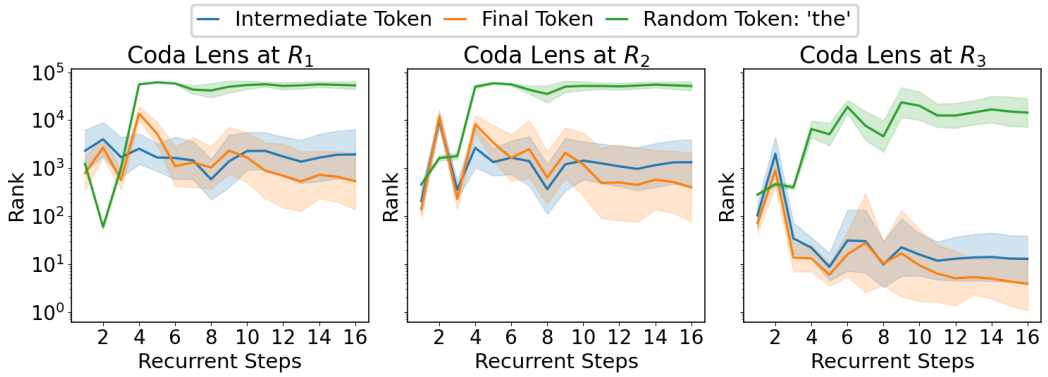

Figure 6: Rank trajectories of the Intermediate, final and random tokens decoded by coda lense at recurrent blocks 1 (left), 2(middle) and 3 (right). Shaded regions denote ± 1 relative std.

A.2 Prompting with Suppressed CoT

4-shot Prompt for section 3.2

You are a concise and helpful assistant. Always return only the final answer straightway.
Question: What is (9 + 8) * 2?
Answer: 34
Question: What is (4 - 7) - 3?
Answer: -6
Question: What is (1 - 5) - 6?
Answer: -10
Question: What is (1 - 9) * 5?
Answer: -40

4-shot Prompt for section 3.3

You are a concise and helpful assistant. Always return only the final answer straightway.
Question: What is (5 + 1) + 1?
Answer: 7
Question: What is (2 + 5) - 1?
Answer: 6
Question: What is (6 - 4) + 5?
Answer: 7
Question: What is (2 + 4) - 1?
Answer: 5

8-shot Prompt for section 3.4

You are a concise and helpful assistant. Always return only the final answer straightway.
There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
6
If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
5
Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
39
Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
8
Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
9
There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
29
Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
33
Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
8

For experiments in section 3.2, section 3.3 and section 3.4, we use the above prompts to guide the model to generate the answer without explicit CoT reasoning. For section 3.3, we

additionally constrain the in-context examples to have single-digit answers, consistent with its experimental setup.

A.3 Further Examples of the Decoded Top-5 Tokens

Logit lens:

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘ ’ , ‘\t’, ‘```’ , ‘ ’ , ‘###’}, {‘ ’ , ‘\t’, ‘```’ , ‘ ’ , ‘###’}, {‘ ’ , ‘\t’, ‘```’ , ‘ ’ , ‘###’}
R_2	{‘```’, ’\t’, ‘###’, ‘##’, ‘ ’}, {‘```’, ’\t’, ‘###’, ‘##’, ‘ ’}, {‘```’, ’\t’, ‘###’, ‘##’, ‘ ’}
R_3	{‘```’, ‘##’, ‘###’, ‘\t’, ‘####’ }, {‘```’, ‘##’, ‘###’, ‘\t’, ‘####’ }, {‘```’, ‘##’, ‘###’, ‘\t’, ‘####’ }
R_4	{‘ heavier’, ‘ colleges’, ‘ coloured’, ‘akis’, ‘ash’}, {‘ heavier’, ‘ colleges’, ‘akis’, ‘ coloured’, ‘ash’}, {‘ heavier’, ‘ colleges’, ‘akis’, ‘ash’, ‘ni’}

Table 2: Top-5 decoded tokens via Logit Lens at recurrent step 1

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘3’, ‘ gre’, ‘5’, ‘2’, ‘1’}, {‘TV’, ‘3’, ‘5’, ‘A’, ‘ tru’}, {‘5’, ‘3’, ‘TV’, ‘MV’, ‘None’}
R_2	{‘3’, ‘5’, ‘2’, ‘ gre’, ‘8’}, {‘3’, ‘TV’, ‘5’, ‘6’, ‘ gre’}, {‘5’, ‘3’, ‘TV’, ‘ inc’, ‘None’}
R_3	{‘3’, ‘2’, ‘8’, ‘5’, ‘1’}, {‘3’, ‘TV’, ‘6’, ‘5’, ‘2’}, {‘5’, ‘3’, ‘6’, ‘TV’, ‘ unity’}
R_4	{‘ weekend’, ‘TED’,‘得’,‘uru’, ‘ gre’}, {‘ines’, ‘ hav’, ‘fly’, ‘ classrooms’, ‘ tru’}, {‘ stal’, ‘igne’, ‘ hav’, ‘off’, ‘ines’}

Table 3: Top-5 decoded tokens via Logit Lens at recurrent step 8

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘3’, ‘2’, ‘1’, ‘None’, ‘|’} {‘TV’, ‘3’, ‘chi’, ‘5’, ‘ spa’} {‘5’, ‘3’, ‘1’, ‘ answer’, ‘None’}
R_2	{‘3’, ‘2’, ‘5’, ‘|’, ‘1’} {‘3’, ‘5’, ‘TV’, ‘chi’, ‘6’} {‘5’, ‘3’, ‘6’, ‘ unity’, ‘ answer’}
R_3	{‘2’, ‘3’, ‘8’, ‘1’, ‘5’} {‘3’, ‘5’, ‘6’, ‘chi’, ‘2’} {‘5’, ‘3’, ‘ unity’, ‘6’, ‘1’}
R_4	{‘ optics’, ‘ decor’, ‘ doctors’, ‘ po’, ‘ chores’} {‘chi’, ‘ani’, ‘Factor’, ‘ hav’, ‘lag’} {‘ inc’, ‘ unity’, ‘ friendships’, ‘igne’, ‘ impulse’}

Table 4: Top-5 decoded tokens via Logit Lens at recurrent step 16

Coda Lens:

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘ plasma’, ‘ sounds’, ‘ draft’, ‘ functions’, ‘```’}, {‘ plasma’, ‘ sounds’, ‘ draft’, ‘```’, ‘ functions’}, {‘ plasma’, ‘ sounds’, ‘ draft’, ‘```’, ‘Answer’}
R_2	{‘Answer’, ‘-’, ‘**’, ‘ plasma’, ‘```’}, {‘Answer’, ‘-’, ‘**’, ‘```’, ‘ plasma’}, {‘Answer’, ‘-’, ‘**’, ‘```’, ‘ plasma’}
R_3	{‘Answer’, ‘```’, ‘-’, ‘**’, ‘####’}, {‘```’, ‘Answer’, ‘-’, ‘**’, ‘####’ }, {‘Answer’, ‘-’, ‘```’, ‘**’, ‘You’ }
R_4	{‘-’, ‘Question’, ‘Your’, ‘Answer’, ‘The’}, {‘-’, ‘Question’, ‘Your’, ‘Answer’, ‘The’}, {‘-’, ‘Question’, ‘Your’, ‘Answer’, ‘The’}

Table 5: Top-5 decoded tokens via Coda Lens at recurrent step 1

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘ answer’, ‘ answering’, ‘ spa’, ‘ Answers’, ‘ answers’}, {‘ answer’, ‘ greeting’, ‘ spa’, ‘ tru’, ‘ product’}, {‘ answer’, ‘ Answer’, ‘ answering’, ‘ highlighting’, ‘ unity’}
R_2	{‘ answer’, ‘ answering’, ‘ answers’, ‘ Answers’, ‘ greeting’}, {‘ answer’, ‘ answering’, ‘ greeting’, ‘ Answer’, ‘ product’}, {‘ answer’, ‘ answering’, ‘ Answer’, ‘ unity’, ‘ answers’}
R_3	{‘Answer’, ‘<|end_turn|>’, ‘ answer’, ‘3’, ‘8’}, {’3’, ’6’, ’Explanation’, ’\\boxed’, ’5’}, {’Explanation’, ’5’, ’Answer’, ’ answer’, ’\\boxed’}
R_4	{‘1’, ‘4’, ‘2’, ‘8’, ‘6’}, {‘6’, ‘1’, ‘3’, ‘8’, ‘7’}, {‘5’, ‘1’, ‘6’, ‘2’, ‘7’}

Table 6: Top-5 decoded tokens via Coda Lens at recurrent step 8

Recurrent Block	Top-5 Logit Lens Decoded Tokens Examples
R_1	{‘ answering’, ‘ answer’, ‘ optics’, ‘ spa’, ‘ tweets’} {‘ answer’, ‘ spa’, ‘ answers’, ‘ greeting’, ‘ alive’} {‘ answer’, ‘ answers’, ‘ tru’, ‘ clarification’, ‘ spa’}
R_2	{‘ answer’, ‘ answering’, ‘ Answer’, ‘ radicals’, ‘ answers’} {‘ answer’, ‘ answers’, ‘ answering’, ‘ greeting’, ‘ Answer’} {‘ answer’, ‘ answers’, ‘ Answer’, ‘ answering’, ‘ tru’}
R_3	{‘8’, ‘Answer’, ‘4’, ‘9’, ‘1’} {‘3’, ‘6’, ‘7’, ‘4’, ‘8’} {‘5’, ‘ answer’, ‘6’, ‘7’, ‘3’}
R_4	{‘1’, ‘8’, ‘-’, ‘2’, ‘6’} {‘6’, ‘1’, ‘8’, ‘7’, ‘3’} {‘6’, ‘5’, ‘1’, ‘7’, ‘2’}

Table 7: Top-5 decoded tokens via Coda Lens at recurrent step 16

As seen in tables 3, 4, 6 and 7, As shown in tables 3, 4, 6 and 7, the proportion of numeric tokens shifts markedly around R_4, but in opposite directions for the two probing methods: it decreases under the Logit Lens and increases under the Coda Lens. This divergence reinforces our analysis and findings in section 3.2.