Mechanistic Interpretability of Large-Scale Counting in LLMs through a System-2 Strategy

Hosein Hasani, Mohammadali Banayeeanzade, Ali Nafisi, Sadegh Mohammadian, Fatemeh Askari, Mobin Bagherian, Amirmohammad Izadi, and Mahdieh Soleymani Baghshah
Sharif University of Technology

Abstract

Large language models (LLMs), despite strong performance on complex mathematical problems, exhibit systematic limitations in counting tasks. This issue arises from the architectural limits of transformers, where counting is performed across layers, leading to degraded precision for larger counting problems due to depth constraints. To address this limitation, we propose a simple test-time strategy inspired by System-2 cognitive processes that decomposes large counting tasks into smaller, independent sub-problems that the model can reliably solve. We evaluate this approach using observational and causal mediation analyses to understand the underlying mechanism of this System-2-like strategy. Our mechanistic analysis identifies key components: latent counts are computed and stored in the final item representations of each part, transferred to intermediate steps via dedicated attention heads, and aggregated in the final stage to produce the total count. Experimental results demonstrate that this strategy enables LLMs to surpass architectural limitations and achieve higher accuracy on large-scale counting tasks. This work provides mechanistic insight into System-2 counting in LLMs and presents a generalizable approach for improving and understanding their reasoning behavior.

1 Introduction

Counting is a fundamental cognitive operation that underpins a wide range of reasoning tasks, from basic arithmetic to more complex forms of quantitative analysis (Feigenson et al., 2004; Dehaene, 2011). In large language models (LLMs), the ability to count is important for controlled generation such as length-constrained summarization, sequential enumeration, and broader numerical/arithmetic reasoning (Retkowski and Waibel, 2025; Hou et al., 2025; Yang et al., 2024c; Gambardella et al., 2024).

Recent research has provided insights into the counting mechanism of LLMs, demonstrating that

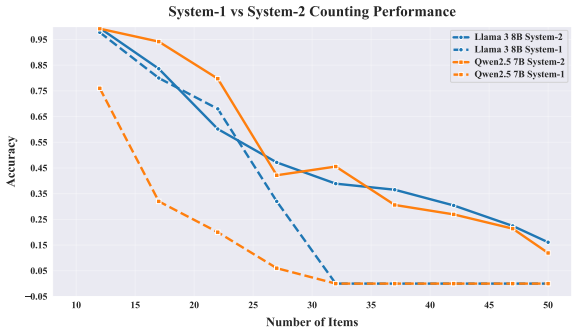


Figure 1: System-1 vs. System-2 counting performance as a function of problem size. System-1 performance degrades rapidly and collapses beyond approximately 30 items, reflecting the bounded capacity of the model's internal counter. In contrast, System-2 counting maintains high accuracy across the entire range by decomposing the task into small solvable sub-problems and aggregating the results.

these models use a layerwise internal counting process where numerical information is progressively accumulated across transformer layers (Hasani et al., 2025b). However, due to the depth limits, this progressive counting mechanism becomes saturated as the number of items increases. Complementary work on numerical representations suggests that LLMs encode numbers in a compressed, sublinear manner, similar to the human logarithmic mental number line (AlquBoj et al., 2025; Dehaene, 2011). While numerical order is preserved, representational resolution decreases with magnitude, explaining the reduced precision for large values.

These findings suggest that LLMs struggle to accurately count large numbers of items, with performance typically degrading for two- and three-digit counts (Zhang et al., 2024; Yehudai et al., 2024; Fu et al., 2024). This limitation reflects a fundamental architectural constraint rather than a lack of training data or supervision, as more layers are required to count a larger number of items (Vaswani et al., 2017; Yehudai et al., 2024; Golkar et al., 2024).

Findings of the Association for Computational Linguistics: ACL 2026, pages 40885–40901

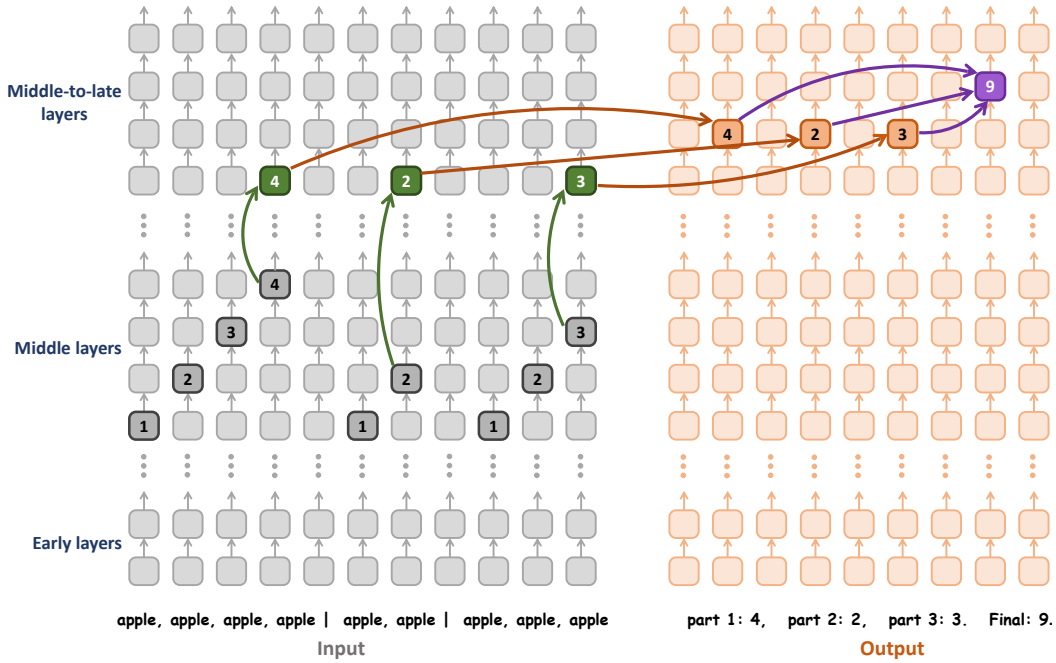


Figure 2: **Internal mechanism of System-2 test-time counting in LLMs.** A large counting task is divided into smaller partitions using an external separator (|). Within each partition, the model performs implicit System-1 counting, where count information accumulates token-by-token and is localized at the final item or separator token (gray blocks). The final count information is transferred via residual streams (green arrows) and stored in the middle-to-late layers (green blocks). These partition-level counts (e.g., 4, 2, 3) are then transferred (orange arrows) through attention pathways to explicit reasoning tokens that report intermediate results (orange blocks). Finally, the intermediate counts are aggregated (purple arrows) to produce the final answer. By keeping each sub-task within the model's reliable counting range, this System-2 procedure removes the upper bound imposed by the model's architectural limitations.

We argue that this failure stems from the model's reliance on a System-1–like processing approach, which is fast, automatic, and capacity-limited (Kahneman, 2011; Dehaene, 2011).

To address this issue, we propose a simple test-time strategy that adopts a System-2–like approach to counting (Figure 1) (Kahneman, 2011). Instead of relying on the model's internal counting mechanisms, which are constrained by architectural limits, our approach decomposes large counting tasks into smaller, independent sub-tasks (Radhakrishnan et al., 2023; Qharabagh et al., 2024). Each sub-task, containing a manageable number of items, can be reliably counted by the model. The results from each sub-task are then aggregated to produce the final count. This approach mirrors human cognitive strategies, where complex problems are broken down into simpler, easier-to-solve sub-problems (Dehaene, 2011). In this study, we use System-1 and System-2 as high-level operational abstractions inspired by human cognition (Kahneman, 2011). Specifically, we refer to System-1 as the model's single-pass, layer-wise processing, and to System-2 as a more deliberate procedure that extends computation across token generation beyond a fixed layer-wise computation.

Our behavioral experiments on various LLMs demonstrate the effectiveness of this strategy in overcoming architectural limitations without requiring model modifications or fine-tuning. Additionally, we provide a detailed mechanistic interpretation of how it functions within LLMs. Using attention analysis and causal mediation techniques (Heimersheim and Nanda, 2024; Geiger et al., 2021; Ghandeharioun et al., 2024; Zhang and Nanda, 2024), we trace the flow of numerical information across the model and identify the mechanisms mediating the System-2 counting process. Figure 2 provides an overview of the main components of the System-2 counting mechanism and its internal information flow. This work offers a new perspective on both improving and understanding LLMs' reasoning capabilities, with a focus on a fundamental cognitive task.

2 Problem Setup and Methodology

We follow the standard counting framework used in prior research (Hasani et al., 2025b). In this setup, a list of repeated items, such as fruits or animals, is presented to the model, and the task is to report the total number of items. For example, given a context like "apple, apple, apple, . . . ", the model must output the total number of items in the list. Previous work has shown that LLMs exhibit high accuracy for small counts (fewer than 10 items), but that performance deteriorates as the number of items increases beyond this range (Zhang et al., 2024; Fu et al., 2024). This suggests that the model's counting ability is limited by the depth of the transformer architecture and its internal counting mechanism (Yehudai et al., 2024).

To overcome this limitation, motivated by prior work on partitioning images in visual reasoning tasks (Izadi et al., 2025), we introduce a simple strategy that explicitly partitions the input list into smaller sub-problems. We use an external separator (|) to divide the list into smaller partitions. For instance, a structured context with three partitions is: "apple, apple, apple, apple, apple | apple, apple, apple | apple, apple, apple, apple, apple, apple".

The model is then instructed to first count the items in each partition and to aggregate the partial counts to produce the final result. This ensures that each sub-problem remains within the model's reliable counting regime. The number of items in each partition is chosen randomly from a range that the model can handle accurately. Models with deeper architectures can reliably process larger partitions. This strategy is based on test-time scaling by leveraging the LLM's inherent capabilities and does not require fine-tuning or any external tools.

Based on the input structure and output format, we consider four baselines in our study. Two input formats are used: an unstructured context with comma-separated items and a structured context with partitions separated by vertical bar symbol (|). For generation, we evaluate two output variants: a short-answer format, where only the final count is produced (corresponding to an immediate System-1-like process), and a Chain-of-Thought (CoT) (Wei et al., 2022) format, where intermediate reasoning steps are included before the final answer (corresponding to a System-2-like process). Full details of the input formats, partitioning, and prompting strategies are provided in Appendix A.1.

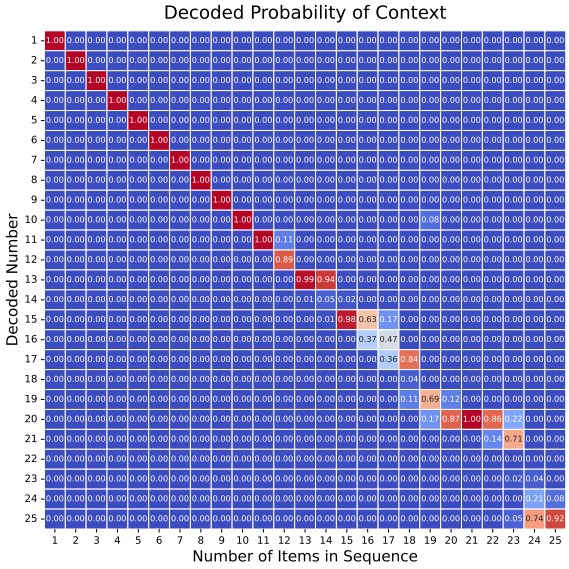


Figure 3: Decoded output probabilities for the unstructured baseline method on Qwen2.5-7B. The heatmap shows the decoded probabilities of model outputs, averaged over different item types, for target counts ranging from 1 to 25. As the count increases beyond 10, the diagonal entries gradually fade, indicating reduced model confidence.

3 Behavioral Results

To illustrate how LLMs lose counting precision in long contexts, we first measure the average prediction probability of Qwen2.5-7B (Yang et al., 2024a) across different context sizes. As shown in Figure 3, model confidence decreases as the number of items exceeds 10. We also observe systematic biases toward more frequent numbers. For example, target counts of 16 and 24 are often predicted as 15 or 25, with 21 and 22 typically predicted as 20.

We then conduct more systematic experiments on both open-source and closed-source models, covering different model sizes, depths, training strategies, and tokenization schemes. For open-source models, we evaluate Qwen2.5-7B, Llama3-8B (Grattafiori et al., 2024), and Gemma3-27B (Kamath et al., 2025), which have 28, 32, and 62 layers, respectively. In addition, we evaluate GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team, 2025) as stronger proprietary models on longer contexts [1]. The corresponding results are reported in Tables 1 and 2. Overall, performance decreases as context size increases and is associ-

[1]For these stronger closed-source models, performance is near-perfect for shorter contexts (below 50 items). We therefore use longer contexts, where the task is no longer saturated and differences between strategies become visible.

Model	Input	Output	Accuracy				MAE			
			11-20	21-30	31-40	41-50	11-20	21-30	31-40	41-50
Qwen2.5-7B	Unstructured	w/o steps	0.38	0.13	0.06	0.00	0.88	2.19	5.29	10.50
		w/ steps	0.45	0.11	0.03	0.00	0.72	2.44	5.97	9.68
	Structured	w/o steps	0.20	0.13	0.05	0.01	3.56	3.54	4.33	6.35
		w/ steps	**0.95**	**0.61**	**0.38**	**0.24**	**0.07**	**1.36**	**1.53**	**2.18**
Llama3-8B	Unstructured	w/o steps	0.80	0.49	0.02	0.00	**0.20**	**0.65**	4.93	11.92
		w/ steps	0.29	0.34	0.00	0.00	0.74	0.82	5.00	11.44
	Structured	w/o steps	0.08	0.05	0.03	0.01	6.62	5.75	5.96	6.62
		w/ steps	**0.84**	**0.54**	**0.38**	**0.26**	0.30	0.70	**1.21**	**2.20**
Gemma3-27B	Unstructured	w/o steps	**1.00**	0.70	0.40	0.00	**0.00**	**0.30**	**1.00**	4.90
		w/ steps	**1.00**	0.50	0.30	0.00	**0.00**	0.60	1.40	4.70
	Structured	w/o steps	0.30	0.05	0.00	0.17	2.10	10.95	7.33	12.67
		w/ steps	**1.00**	**0.85**	**0.55**	**0.50**	**0.00**	0.35	2.15	**2.25**

Table 1: Average accuracy and mean absolute error (MAE) across **open-source models** for context sizes from 11 to 50. Each model is evaluated on structured and unstructured inputs, with and without intermediate reasoning steps. MAE complements exact-match accuracy by measuring how far incorrect predictions are from the true count.

Model	Input	Output	Accuracy					MAE				
			51-60	61-70	71-80	81-90	91-100	51-60	61-70	71-80	81-90	91-100
GPT-4o	Unstructured	w/o steps	0.70	0.54	0.56	0.18	0.24	0.36	1.42	0.72	3.62	4.26
		w/ steps	0.58	0.53	0.40	0.16	0.26	1.10	1.78	1.72	3.20	3.64
	Structured	w/o steps	0.37	0.31	0.10	0.11	0.11	1.04	1.53	3.22	2.64	3.03
		w/ steps	**0.96**	**0.91**	**0.87**	**0.83**	**0.86**	**0.04**	**0.10**	**0.16**	**0.22**	**0.18**
Gemini-2.5-Pro	Unstructured	w/o steps	0.52	0.50	0.44	0.42	0.20	0.60	0.50	3.17	4.67	2.70
		w/ steps	0.95	0.80	0.72	0.60	0.60	**0.05**	1.10	1.78	1.30	1.30
	Structured	w/o steps	0.82	0.80	0.78	0.75	0.79	0.25	0.08	0.18	0.30	0.10
		w/ steps	**0.97**	**0.95**	**0.95**	**0.91**	**0.91**	0.10	**0.05**	**0.05**	**0.06**	**0.07**

Table 2: Average accuracy and MAE across **closed-source models** for context sizes from 51 to 100. Each model is evaluated on structured and unstructured inputs, with and without intermediate reasoning steps.

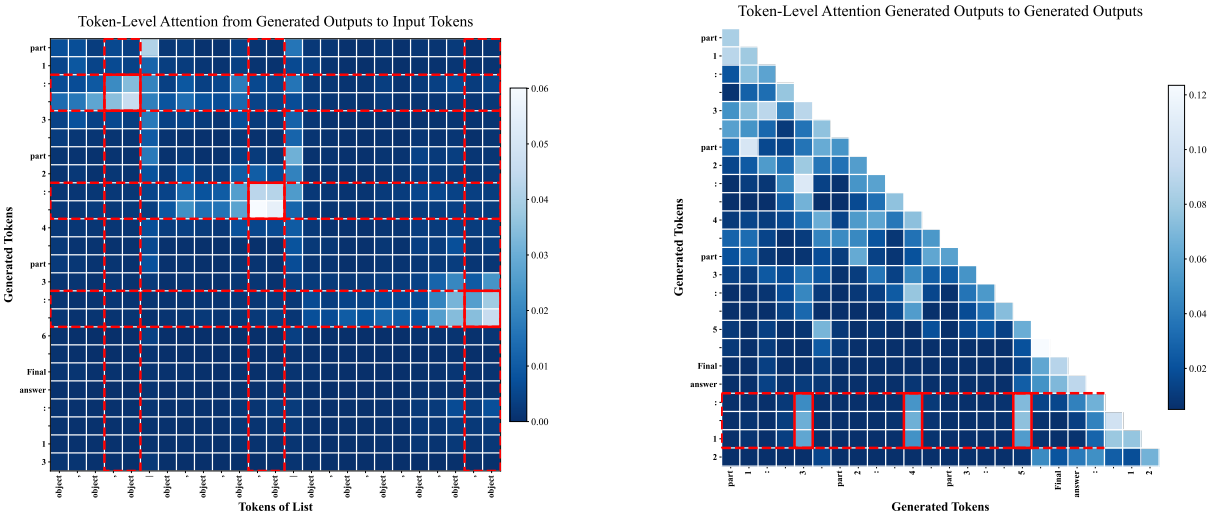
ated with model scale, especially depth, with larger and deeper models showing higher accuracy. For each model, we find that only one configuration consistently succeeds on long contexts: structured inputs combined with intermediate reasoning steps.

Surprisingly, encouraging CoT reasoning alone, without structured input, does not yield a notable improvement. Furthermore, structured input without intermediate steps is also ineffective and can be harmful in some cases. These results show that neither external structure nor reasoning alone is sufficient. Their combination is necessary to overcome large-scale counting failures. In many models, the unstructured short-answer baseline performs better than the structured short-answer baseline. This suggests that the model must first consolidate partial results in intermediate steps before aggregating them into the final answer through a two-stage process.

A likely explanation for the failure of the structured short-answer setting is related to the latent counting mechanism. Prior work showed that some models tend to output the maximum latent count observed in the context rather than the true total count (Hasani et al., 2025b). In our structured format, the counter resets after each partition, so the maximum latent count often matches the largest partition size instead of the total sum. Consistent with this view, for context sizes 11 to 30, 13% of Qwen2.5-7B errors and 43.6% of Llama3-8B errors exactly match the largest partition size. In addition, the high MAE of this setting in Table 1 suggests systematic bias in erroneous predictions. Our observational and causal analyses provide further insight into why models cannot simultaneously gather partial counts from the context and add them up without loss.

Finally, we analyze failure cases of the structured CoT setting by separating errors into intermediate partition counts and final aggregation. Table 3 shows that the main source of failure is the intermediate steps. This indicates that once correct partition counts are produced, most models can reliably add them.

Additional experiments on smaller and math-specialized models are presented in Appendix A.2, showing that the observed trends generalize across model scale and domain specialization. Robustness checks across tokenizers and alternative input structures are provided in Appendix A.3, and the



(a) Attention of generated tokens to input tokens.

(b) Attention of generated tokens to generated tokens.

Figure 4: Attention patterns of selected tokens under the System-2 counting strategy. Attention values are averaged across layers 19 to 23, all heads, and all item types (e.g., different fruit names).

Model	Total Acc	Final Step Acc	Interm. Acc
Qwen2.5 7B	0.51	0.86	0.53
Llama 3 8B	0.49	0.96	0.48
Gemma 3 27B	0.71	0.93	0.76
GPT-4o	0.89	1.00	0.89
Gemini-2.5-Pro	0.94	0.97	0.94

Table 3: Failure analysis of the structured CoT setting. Total accuracy measures exact final answers. Final-step accuracy measures correctness of the aggregation step conditioned on generated intermediate counts. Intermediate-step accuracy measures correctness of partition-level counts.

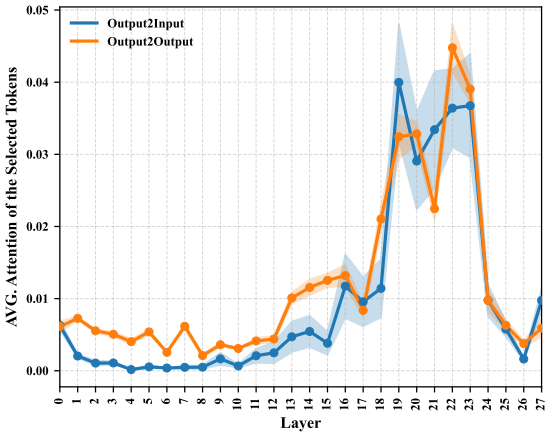


Figure 5: Average attention of correctly generated intermediate counts to the final item of their partition, alongside attention of the final answer to partition-level counts, across layers. Higher attention values are observed from layer 19 to 23 for both paths.

overall trends hold across tokenizers and input formats, demonstrating that the System-2 mechanism is robust and generalizable.

4 Attention Analysis

We analyze attention patterns of Qwen2.5-7B (Yang et al., 2024a) to understand how the proposed System-2 strategy enables large-scale counting. We first identify which tokens are attended to when the model generates (i) intermediate partition-level counts and (ii) the final aggregated answer. We then localize the layers and heads that contribute most to these processes.

Figure 4a shows attention patterns to the input items when the model generates intermediate reasoning steps. We consider a structured context with three partitions containing 3, 4, and 6 items. During the generation of a reasoning token such as "part 2: 4", the attention weight of the generated number peaks sharply on the final item and the final comma separator of the corresponding partition. This pattern is consistent across prompts and is most pronounced in middle to late layers, typically around layers 19 to 23. Earlier items within the same partition receive substantially lower attention.

Figure 4b shows attention patterns when the model generates the final answer. The ":" token and the whitespace token immediately preceding the final number attend strongly to all intermediate reasoning numbers produced earlier (e.g., 3, 4, and 5). The attention mass is distributed across these tokens, with a clear concentration in the same layers identified for intermediate steps. Attention to

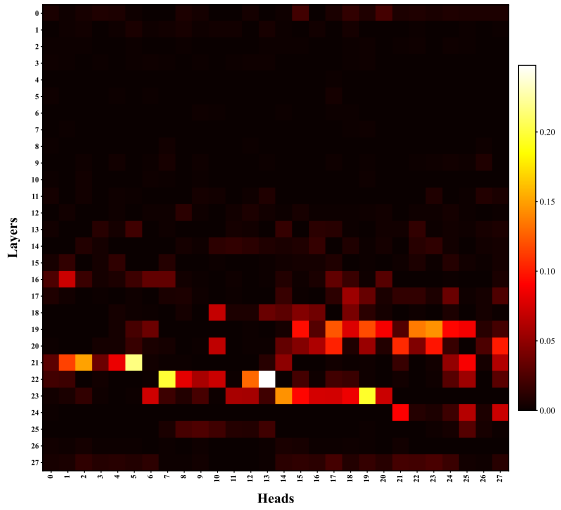


Figure 6: Heatmap of average attention to final items of partitions across layers and heads for intermediate counting steps. Layers and heads responsible for transferring numerical information from input partitions to intermediate counts appear with higher intensity.

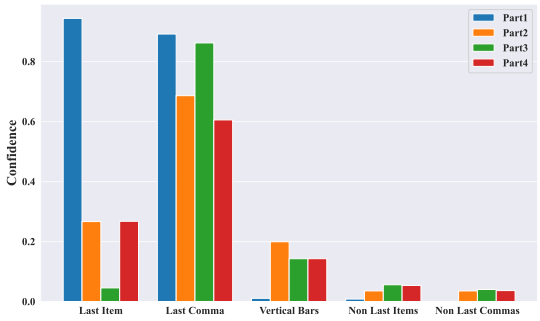


Figure 7: Ground-truth probabilities across different tokens of partitions decoded by CountScope. Probabilities are averaged over different item types and configurations (3 to 9 items per partition).

the original input items is weak at this stage.

To further localize the attention pathways involved in intermediate reasoning and final aggregation, we focus on the most relevant tokens identified in Figure 4. We average attention values across different configurations, including item types and partition sizes. Figure 5 shows that attention peaks in layers 19 to 23, highlighting the central role of these layers in information transfer and aggregation. In addition, Figure 6 further shows that the most influential heads are concentrated in layers 21, 22, and 23. For example, head 13 in layer 22 consistently exhibits high attention to the selected tokens. Additional methodological details and attention analyses across multiple model architectures are provided in Appendix A.4.

Together, these results suggest a staged computation. First, each partition is counted independently, with the final tokens of the partition encoding the local count. Second, these local counts are written into intermediate reasoning tokens. Finally, the model attends to these intermediate tokens and aggregates their values to produce the final answer. These observations are consistent with a hypothetical mechanism that separates counting, information transfer, and aggregation into distinct components.

5 Causal Mediation Analysis

To assess the hypothesis of a multi-stage System-2 counting mechanism suggested by the attention analysis, we perform a set of causal mediation experiments based on activation patching (Heimersheim and Nanda, 2024; Zhang and Nanda, 2024), masking ablation, and attention knockout (Geva et al., 2023). The goal is to identify where partition-level count information is stored, how it is transferred and consolidated, and which components are causally required for correct aggregation.

5.1 Token-Level Information Probing

We first examine where partition-level counts are encoded in the input context. This analysis requires vocabulary projection and probing tools. We observe that existing tools, such as logit-lens (NostAlgebraist, 2020) and tuned-lens (Belrose et al., 2023), are not reliable for decoding numerical information. We therefore use the CountScope method (Hasani et al., 2025b) to decode the implicit count associated with tokens. CountScope is a causal probing method that patches the activation of a target token into a blank counting context and decodes the model's output as the implied count.

Figure 7 shows the average decoding probability of ground-truth numbers across different token types. Ground-truth numbers correspond to the number of items in each partition. The results show that the ground-truth value is encoded with high confidence primarily at the final item and the final comma separator of each partition. For the first partition, the confidence at the final item is high; however, for subsequent partitions, the confidence at this token decreases, while the confidence at the vertical bar character increases. Nevertheless, the final comma separator reliably stores the latent count of the corresponding partition. These findings are consistent with the attention results in Section 4, which showed that decoded numbers of intermediate steps attend most strongly to these

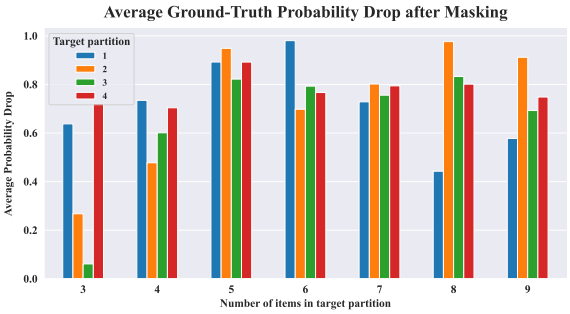


Figure 8: Average ground-truth probability drop after masking the final item and comma (e.g., ‘, apple’) from each partition, showing the effect on the target count for digit sizes ranging from 3 to 9.

final partition tokens.

More interestingly, this experiment reveals that at each partition boundary, the count resets and begins again. As a result, the final item of each partition encodes the number of items counted since the beginning of that partition. This observation explains why the System-2 strategy avoids the large-number failure observed in unstructured inputs. Each partition is counted independently, and its size remains within the range where the model's implicit counter is accurate.

5.2 Token-Level Causal Interventions

To test whether the identified tokens causally mediate the model's behavior, we perform zero ablation on the final items and separators of each partition. Specifically, after processing the entire input sequence and extracting token embeddings, we replace the activation values of selected tokens with zeros across all layers.

Figure 8 shows the average probability drop after zero ablation of the final item and final comma separator. This intervention leads to a sharp drop in the probability of generating the correct partition-level count in the corresponding reasoning step. For example, ablating the final “, apple” token of a four-item partition significantly reduces the likelihood of generating the token “4”.

5.3 Information Pathway Localization

To perform fine-grained circuit localization, we analyze the pathways responsible for System-2 counting using attention knockout (Geva et al., 2023). We selectively block individual attention heads and layers and measure the resulting drop in counting accuracy for intermediate and final steps. Figure 9 shows that for Qwen2.5-7B, the most influential components are concentrated in layer 22. In particular, attention head 13 is most important for intermediate steps, while head 1 is most important for the final aggregation step (Figure 10).

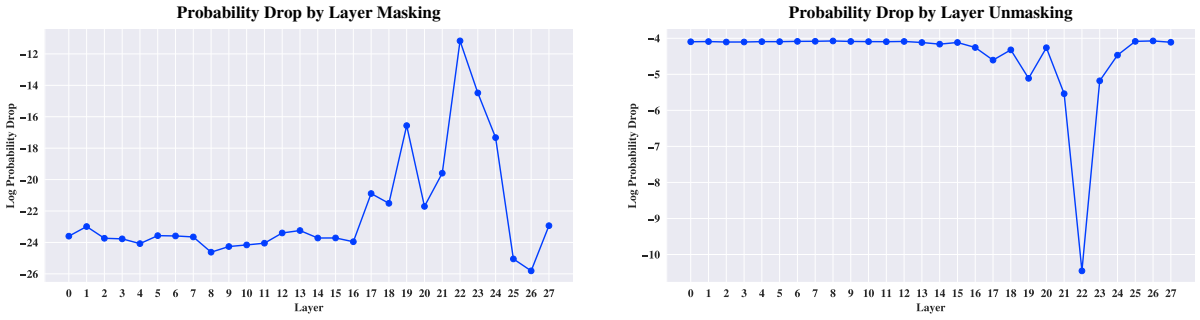
Notably, head 13 in layer 22 also exhibits the highest attention values in the attention heatmap shown in Figure 6. Compared to Figure 6, the pattern in Figure 10 is considerably sparser. A plausible explanation is the presence of parallel information pathways, where knocking out a single pathway does not fully disrupt the computation because other pathways can partially compensate. Finally, we observe that the heads responsible for transferring partition-level information from the input context to the intermediate steps are not necessarily the same as those responsible for transferring information from the intermediate reasoning steps to the final answer, even when they are located in the same layer.

5.4 Cross-Context Activation Patching

Finally, we perform cross-context activation patching to test how partition-level information is combined to form the final answer. We focus on intermediate responses and final aggregation steps. Our attention and masking analyses indicate that transferred numerical information from the context to intermediate steps is consolidated in the specific tokens “:”, the whitespace token, and the partition number. Here, we investigate this behavior using a causal intervention setup (Geiger et al., 2021).

To this end, we sample two different contexts, each containing four partitions, with partition sizes randomly chosen between 3 and 9. After generating the intermediate steps, we select one partition and transfer the embeddings from layers 18 to 24 of the target tokens between the two responses. For example, let “part 1: 7, part 2: 4, part 3: 8” be the intermediate steps of the first context and “part 1: 5, part 2: 6, part 3: 3” be those of the second context. We swap the intermediate activations of the tokens highlighted in blue between the two responses. After this intervention, we allow the model to generate the final response.

This manipulation causally affects the corresponding intermediate steps and changes the final count accordingly. In the given example, the total sums of the first and second contexts are 19 and 14 before activation patching. After swapping the numerical contents of their second steps (shown in blue), the total for the first context becomes 21 and the total for the second context becomes 12. This



(a) Masking (Deletion)

(b) Unmasking (Insertion)

Figure 9: Average change in log probability of intermediate counts after layer masking (deletion) and unmasking (insertion) of the final item and separator of each partition. In the masking experiment, embeddings from the target layer are zero-ablated. In the unmasking experiment, embeddings of the selected tokens are zero-ablated across all layers, and only the target-layer embeddings are restored from the clean run.

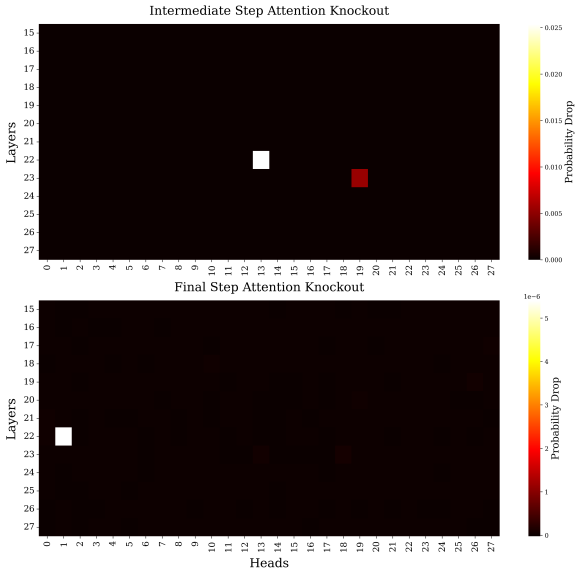


Figure 10: Heatmap of the average probability drop after attention knockout across layers and heads for intermediate (top) and final counting steps (bottom). Attention knockout is applied only to the identified effective tokens from the input context and intermediate steps.

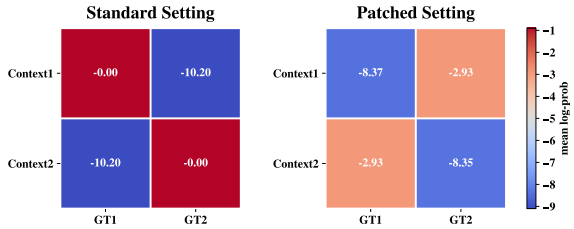


Figure 11: Average log probabilities of the first and second contexts before (left) and after (right) activation patching. The first and second columns correspond to the predicted outputs (total sums) of the first and second contexts. After swapping the layer embeddings of the selected tokens from a given partition between the two contexts, the output no longer follows the original total sum and instead reflects the transferred number. Values are averaged across different configurations (selected partitions, partition sizes, and item types).

shows that the final sum is causally mediated by the intermediate-layer embeddings of the tokens ":", whitespace, and the partition number. Figure 11 shows the log probabilities for this experiment, averaged over various configurations.

6 Related Work

Counting in LLMs. A growing body of work shows that counting in LLMs is brittle: performance can hinge on surface form and segmentation rather than a stable procedure (Fu et al., 2024). Subword tokenization can blur item boundaries and measurably affect counting accuracy (Zhang et al., 2024). More broadly, deterministic-task evaluations (including counting) can flip under minor prompt/content changes, complicating extrapolation from narrow setups (Ball et al., 2024). On the theory side, transformers can exactly count token frequencies only in specific regimes, with feasibility tied to representation capacity and context-length scaling (Yehudai et al., 2024).

Numerical representations and mechanistic views. Cognitive work argues for specialized number systems and structured magnitude representations (Feigenson et al., 2004; Dehaene, 2011), motivating analogous questions in neural models. Representation analyses study how numerical magnitude is organized in LLM hidden spaces and relate these structures to human-like effects (AlquBoj et al., 2025). Mechanistic studies localize how counting signals emerge and update across items

in controlled settings (Golkar et al., 2024). Closest to our setting, Hasani et al. (2025b) combine a target-context probe with layerwise/token-level analyses and causal interventions to identify latent counter-like signals across both LLMs and LVLMs.

Decomposition and reasoning traces. Chain-of-Thought can improve multi-step reasoning by eliciting intermediate computations (Wei et al., 2022), but explicit traces are not necessarily faithful. Question decomposition work studies when decomposition improves faithfulness rather than mainly serving as scaffolding (Radhakrishnan et al., 2023). Decomposing images into smaller regions has also been shown to be effective for visual reasoning (Izadi et al., 2025; Hasani et al., 2025a). This strategy improves visual counting performance (Izadi et al., 2025; Qharabagh et al., 2024; Hasani et al., 2025a), but its underlying mechanism remains underexplored.

Interpretability tools. Causal interventions such as activation patching help identify computation-critical states (Heimersheim and Nanda, 2024; Geiger et al., 2021), with best-practice guidance for reliable metrics and methods (Zhang and Nanda, 2024). Patch-based inspection frameworks further systematize intervention/inspection configurations (Ghandeharioun et al., 2024). Readout-style probes provide a complementary perspective by tracking when information becomes linearly decodable across layers, including the Logit Lens (NostAlgebraist, 2020) and learned variants such as the Tuned Lens (Belrose et al., 2023).

Positioning of This Work. Our setup is closest to System-2-style prompting via intermediate steps: like CoT, we elicit explicit intermediate computations (Wei et al., 2022), and like task decomposition, we solve the instance by breaking it into sequential subproblems (Radhakrishnan et al., 2023). The key difference is that the decomposition is tied to a *structured input format*: following prior vision-based studies (Izadi et al., 2025; Hasani et al., 2025a), we first partition the input text into marked segments and then use a CoT protocol aligned with this structure by producing segment-level subcounts and then aggregating them. This creates explicit stage boundaries that are absent in monolithic counting prompts and lets us study how counting representations evolve across stages. Mechanistically, our analysis aligns with prior work that localizes latent counter-like signals and tests them with interventions (Hasani et al., 2025b), while shifting the question to how the *structured CoT decomposition* reshapes internal computation on long contexts. In this way, we connect behavioral brittleness observations (Ball et al., 2024; Zhang et al., 2024) to a stage-wise mechanistic account of how count information is formed, routed, and combined under structured prompting (Heimersheim and Nanda, 2024; Zhang and Nanda, 2024).

7 Conclusion

This paper showed that the failure of large language models on large-scale counting tasks arises from reliance on System-1–like implicit counting mechanisms with limited capacity (Kahneman, 2011; Zhang et al., 2024). This problem reflects architectural constraints of transformer models rather than fundamental limits of numerical reasoning. We introduced a simple System-2 test-time strategy that decomposes large counting problems into smaller sub-tasks and aggregates their results, enabling accurate counting over long contexts without modifying model parameters or training procedures.

In addition to behavioral improvements, we provided a mechanistic explanation of how this strategy operates internally. Using attention analysis and causal mediation methods (Hasani et al., 2025b; Heimersheim and Nanda, 2024), we showed that partition-level counts are encoded at boundary tokens, transferred through specific attention pathways to intermediate reasoning steps, and aggregated in middle-to-late layers to form the final answer. Intermediate reasoning tokens play a crucial role in this process, mediating the flow of numerical information from input partitions to the final output.

In principle, this procedure removes a fixed upper bound on countable size, as long as each sub-problem remains within the model's reliable regime. While our experiments focus on counting, the same analysis framework may extend to other reasoning tasks where implicit representations saturate and explicit decomposition into sub-tasks facilitates correct behavior. This study highlights how structured test-time strategies can reveal and extend the computational abilities of existing models, offering a path toward both improved performance and deeper interpretability. Future research could explore further applications of this strategy to more complex reasoning tasks in language and beyond multimodal settings.

Limitations

This work focuses on a text-based counting task with limited object diversity. For systematic mechanistic analysis, we use synthetic contexts with repeated nouns rather than natural free-form text. The approach depends on structured prompts and assumes prior knowledge of the model's reliable counting range. While this requirement is mild in practice, it may limit applicability across domains. The proposed decomposition strategy is most effective for tasks that can be divided into near-independent sub-tasks (e.g., multi-step arithmetic or sequential planning). It is not expected to generalize to tasks with strong interdependencies, where subproblems cannot be cleanly separated.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, and et al. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

H. V. AlquBoj, Hilal AlQuabeh, Velibor Bojkovic, Tatsuya Hiraoka, Ahmed Oumar El-Shangiti, Munachiso Nwadike, and Kentaro Inui. 2025. Number representations in llms: A computational parallel to human perception. In *International Conference on Learning Representations (ICLR)*.

Thomas Ball, Shuo Chen, and Cormac Herley. 2024. Can we count on llms? the fixed-effect fallacy and claims of gpt-4 capabilities. *arXiv preprint arXiv:2409.07638*.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Stanislas Dehaene. 2011. *The Number Sense: How the Mind Creates Mathematics*, 2nd edition. Oxford University Press, New York, NY.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2026. Implicit chain of thought reasoning via knowledge distillation, 2023. *arXiv preprint arXiv:2311.01460*.

Lisa Feigenson, Stanislas Dehaene, and Elizabeth Spelke. 2004. Core systems of number. *Trends in Cognitive Sciences*, 8(7):307–314.

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? *arXiv preprint arXiv:2412.18626*.

Andrew Gambardella, Yusuke Iwasawa, and Yutaka Matsuo. 2024. Language models do hard arithmetic tasks easily and hardly do easy arithmetic tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Bangkok, Thailand. Association for Computational Linguistics.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *International Conference on Machine Learning (ICML)*.

Siavash Golkar, Alberto Bietti, Mariel Pettee, Michael Eickenberg, Miles Cranmer, Keiya Hirashima, Geraud Krawezik, Nicholas Lourie, Michael McCabe, Rudy Morel, Ruben Ohana, Liam Holden Parker, Bruno Régaldo-Saint Blancard, Kyunghyun Cho, and Shirley Ho. 2024. Contextual counting: A mechanistic study of transformers on a quantitative task. *arXiv preprint arXiv:2406.02585*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. 2025a. Uncovering grounding ids: How external cues shape multimodal binding. *Preprint*, arXiv:2509.24072.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. 2025b. Understanding counting mechanisms in large language and vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. arXiv preprint arXiv:2404.15255.

Kuinan Hou, Marco Zorzi, and Alberto Testolin. 2025. Sequential enumeration in large language models. *Preprint*, arXiv:2512.04727.

Amirmohammad Izadi, Mohammadali Banayeeanzade, Fatemeh Askari, Ali Rahimiakbar, Mohammad Mahdi Vahedi, Hosein Hasani, and Mahdieh Soleymani Baghshah. 2025. Visual structures help visual reasoning: Addressing the binding problem in lvlms. In *Advances in Neural Information Processing Systems*.

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, NY.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, and 1 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

NostAlgebraist. 2020. Interpreting gpt: The logit lens. LessWrong post.

Muhammad Fetrat Qharabagh, Mohammadreza Ghofrani, and Kimon Fountoulakis. 2024. LVLM-COUNT: Enhancing the counting ability of large vision-language models. *arXiv preprint arXiv:2412.00686*.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilē Lukošiūtē, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, and 5 others. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv:2307.11768*.

Fabian Retkowski and Alexander Waibel. 2025. Zero-shot strategies for length-controllable summarization. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 551–572, Albuquerque, New Mexico. Association for Computational Linguistics.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv:2507.06261*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, and 1 others. 2024a. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Haotong Yang, Yi Hu, Shijia Kang, Zhouchen Lin, and Muhan Zhang. 2024c. Number cookbook: Number understanding of language models and how to improve it. *Preprint*, arXiv:2411.03766. ICLR 2025 poster.

Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. 2025. Language models can learn implicit multi-hop reasoning, but only if they have lots of training data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9695–9713.

Gilad Yehudai, Haim Kaplan, Asma Ghandeharioun, Mor Geva, and Amir Globerson. 2024. When can transformers count to n? *arXiv preprint arXiv:2407.15160*.

Fred Zhang and Neel Nanda. 2024. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations (ICLR)*.

Xiang Zhang, Juntai Cao, and Chenyu You. 2024. Counting ability of large language models and impact of tokenization. *arXiv preprint arXiv:2410.19730*.

A Appendix

A.1 Details of the Experimental Setup

We evaluate counting performance under controlled variations of input format and prompting strategy. Each example consists of a list of repeated single-word items, and the model is required to output the total count. Following the evaluation protocol summarized in Tables 1 and 2, we vary (i) whether the input is *structured* or *unstructured*, and (ii) whether the model is explicitly encouraged to produce intermediate reasoning steps (*with steps* vs. *without steps*). Performance is reported using exact-match accuracy and mean absolute error (MAE) across different context-length ranges.

Item types. All inputs are constructed from simple, common nouns drawn from two semantic categories: *fruits* and *animals*. The complete candidate set is shown below.

Items

```
apple, orange, peach, fig, mango,
pear, coconut, cherry, plum,
cat, dog, horse, rabbit, whale, cow,
frog
```

Input formats. In the **unstructured** setting, the input is a flat, comma-separated list of n identical items. In the **structured** setting, the same multiset is divided into multiple partitions separated by a vertical bar (|), enabling explicit local counting followed by summation.

Partition sizes. For **open-source models** (Table 1), each partition contains between **6 and 9 items**. For **closed-source models** (Table 2), which are evaluated on larger context lengths, partition sizes range from **15 to 25 items**. In all cases, partitions are constructed such that their sizes sum exactly to the target length n.

Prompting strategies. For each input format, we evaluate two prompting strategies: *without intermediate steps* and *with intermediate steps*. These settings correspond directly to the "w/o steps" and "w/ steps" rows reported in Tables 1 and 2.

Unstructured Input (w/o steps)

```
You will be given a list of items.
Count the total number of objects.
Output the final result exactly in the
following format:
Final answer: [x]
```

Unstructured Input (w/ steps)

```
You will be given a list of items.
Count the total number of objects.
Let's count step by step.
Output the final result exactly in the
following format:
Final answer: [x]
```

Structured Input (w/o steps)

```
You will be given multiple partitions
of content separated by "|".
For each partition, count the number
of items it contains.
After counting all partitions,
compute the total by summing the
counts.
Output the final result exactly in
the following format:
Final answer: [x]
```

Structured Input (w/ steps)

```
You will be given multiple partitions
of content separated by "|".
For each partition:
- Count the number of items it
contains.
- Report the count separately using
the format:
part1: [x1]
part2: [x2]
...
After counting all partitions:
- Compute the total by summing all
individual counts.
- Output the final total exactly in
this format:
Final answer: [x]
```

Model	Input	Output	11–20	21–30	31–40	41–50
Gemma2-2B	Unstructured	w/o steps	0.01	0.00	0.00	0.00
		w/ steps	0.00	0.00	0.00	0.00
	Structured	w/o steps	0.05	0.12	0.03	0.04
		w/ steps	**0.18**	**0.15**	**0.10**	**0.07**
Gemma3-4B	Unstructured	w/o steps	0.91	0.45	0.04	0.08
		w/ steps	**0.99**	0.52	0.02	0.03
	Structured	w/o steps	0.26	0.13	0.03	0.05
		w/ steps	0.96	**0.56**	**0.19**	**0.17**

Table 4: Average accuracy on **smaller open-source models** (Gemma2-2B and Gemma3-4B) for context sizes from 11 to 50. Each model is evaluated on structured and unstructured inputs, with and without intermediate reasoning steps.

Model	Input	Output	11–20	21–30	31–40	41–50
Qwen2.5-Math-7B	Unstructured	w/o steps	0.95	0.54	0.04	0.04
		w/ steps	0.94	0.46	0.02	0.05
	Structured	w/o steps	0.97	**0.89**	**0.89**	**0.75**
		w/ steps	**1.00**	0.77	0.75	0.52
DeepSeekMath-7B	Unstructured	w/o steps	0.86	0.12	0.04	0.04
		w/ steps	0.90	0.30	0.03	0.05
	Structured	w/o steps	0.40	0.21	0.03	0.01
		w/ steps	**0.86**	**0.58**	**0.38**	**0.19**

Table 5: Average accuracy on **math-specialized models** for context sizes from 11 to 50. DeepSeekMath-7B follows the same trend as general-purpose models, where structured input with intermediate reasoning performs best. Qwen2.5-Math-7B shows stronger performance for both structured settings, suggesting improved internal aggregation of partition-level counts.

A.2 Behavioral Results on Additional Models

We extend the behavioral evaluation to two smaller open-source models, Gemma2-2B (Riviere et al., 2024) and Gemma3-4B (Kamath et al., 2025), and two math-specialized models, Qwen2.5-Math-7B (Yang et al., 2024b) and DeepSeekMath-7B (Shao et al., 2024). We use the same four settings as in the main paper: unstructured vs. structured inputs, each with short-answer generation or intermediate reasoning steps. We evaluate context sizes from 11 to 50 items, grouped into four bins.

Results for smaller models are shown in Table 4. As expected, absolute accuracy decreases with model size, especially for longer contexts. However, the main trend from the paper remains unchanged. Structured input with intermediate reasoning is the strongest setting overall and remains more robust than the other three variants as context length increases. This suggests that the proposed decomposition strategy is not limited to larger models and still provides benefits in lower-capacity regimes.

Results for math-specialized models are shown in Table 5. DeepSeekMath-7B follows the same pattern as general-purpose models, where structured input with intermediate reasoning gives the best performance. Qwen2.5-Math-7B shows a different behavior. Both structured settings perform strongly, and structured input without intermediate reasoning is often competitive with, or better than, the version with reasoning steps. This suggests that some math-tuned models may better combine partition-level counts internally without requiring explicit intermediate outputs. This behavior is also consistent with stronger out-of-context reasoning ability, where some models remain effective in the no-CoT regime (Deng et al., 2026; Yao et al., 2025).

These additional experiments support the main conclusion that external structure is highly useful for long-range counting. For most models, the strongest configuration remains structured input with intermediate reasoning. At the same time, the Qwen2.5-Math result suggests that domain-specific fine-tuning can partially change how aggregation is carried out at inference time while maintaining strong performance in the no-CoT setting.

A.3 Robustness to Tokenization and Input Structure

We evaluate the generality of our System-2 counting mechanism across different tokenization

schemes and input formats. Table 6 summarizes the tokenizer types and vocabulary sizes of the models used in our main experiments. Despite these differences, the overall behavioral and mechanistic patterns are consistent, indicating that the counting strategy does not rely on a specific tokenizer or vocabulary.

Model	Tokenizer Type	Vocabulary Size
GPT-4o	BPE	∼200k
Gemini 2.5 Pro	SentencePiece	∼256k
Gemma 3	SentencePiece	∼262k
Llama 3	BPE	∼128k
Qwen 2.5	Byte-level BPE	∼150k

Table 6: Tokenizer types and vocabulary sizes for the evaluated models.

To further assess robustness, we evaluate two alternative input structures with different separators and instructions:

Structure A: Intermediate structures use part labels with commas separating items, e.g.,

```
part1: apple, apple
part2: apple, apple, apple
```

Structure B: Intermediate structures list items with spaces, and parts are separated by slashes, e.g.,

```
apple apple / apple apple apple
```

Table 7 shows that the same performance trends hold for Qwen2.5-7B: structured inputs combined with intermediate reasoning steps consistently outperform unstructured inputs or inputs without reasoning steps, confirming that the System-2 mechanism is robust to minor changes in input formatting. However, the superior performance of Structure A over Structure B highlights the role of input structure in overall performance.

A.4 Attention Analysis

Input Details The exact prompt structure is provided below:

```
You will be given a list of items, where groups
    (partitions) are separated by the "|"
    character.

For each partition:
- Count the number of items it contains.
- Report the count separately using the format:
  part1: x1
  part2: x2
  part3: x3
  ...

After counting all partitions:
- Compute the total by summing all individual
    counts.
- Output the final total exactly in this format:
Final answer: x
Just answer in this format without any extra
    things and follow the instructions.

apple, apple, apple | apple, apple, apple,
    apple | apple, apple, apple, apple, apple
```

To evaluate robustness across varying input structures, the final list provided to the model is randomized across experiments. Additionally, to reduce potential token-specific biases, we vary the item labels used in the lists, drawing from the Appendix A.1

Further Attention Results We extend the attention analysis to several large language models, including Qwen2.5-7B (Yang et al., 2024a), Llama3.2 8B (Grattafiori et al., 2024), and Gemma3-4B (Kamath et al., 2025). These models differ in architectural depth and layer organization, enabling a more comprehensive comparative analysis of attention behaviors across architectures.

As shown in Figure 12, we analyze attention scores across different layer ranges for each model. For Gemma3-4B, we examine layers 21 to 23, while for Llama3.2-8B we consider layers 13 to 18. These layer intervals are selected based on the average attention magnitude from output tokens to key input tokens (specifically, the last item and the trailing comma of the predicted partition), which consistently peak within these ranges.

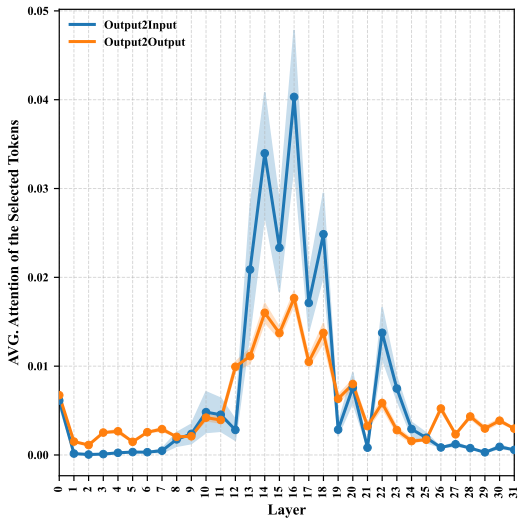
Moreover, Figures 13, 14, and 15 illustrate the full attention maps for Qwen2.5-7B, Llama3.2 8B, and Gemma3-4B, respectively. In each figure, the x-axis corresponds to output tokens and the y-axis corresponds to input tokens. Across all models, we observe a consistent pattern: the token immediately preceding the generated number of parts exhibits the highest attention weights toward the last item and the final comma of the desired partition. This behavior supports our hypothesis that models rely heavily on attention to the final item–comma structure of each part when determining and generating the number of items for each part.

A.5 Causal Mediation Analysis

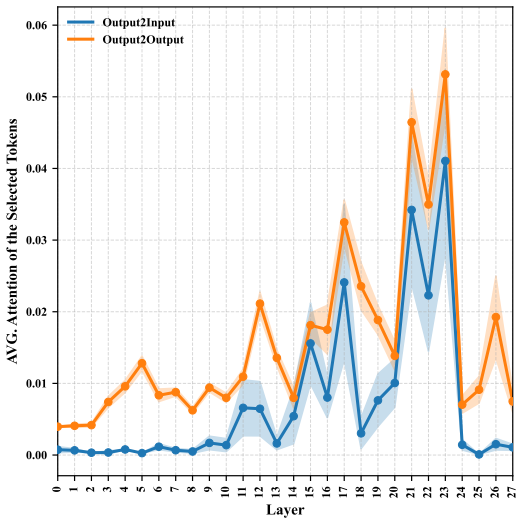
Token-Level Information Probing To decode the latent numbers of specific tokens, we employed CountScope. We modified the original source prompt to facilitate accurate counting by partitioning the input list using vertical bar delimiters (|). Additionally, we utilized a monotypic, question-first configuration. The exact prompt structure is

Structure	Input	Output	11–20	21–30	31–40	41–50
A	Unstructured	w/o steps	0.30	0.11	0.05	0.00
	Unstructured	w/ steps	0.45	0.10	0.02	0.00
	Structured	w/o steps	0.38	0.24	0.16	0.10
	Structured	w/ steps	**1.00**	**0.77**	**0.64**	**0.32**
B	Unstructured	w/o steps	0.28	0.09	0.05	0.01
	Unstructured	w/ steps	0.28	0.10	0.04	0.01
	Structured	w/o steps	0.22	0.19	0.12	0.10
	Structured	w/ steps	**0.74**	**0.46**	**0.39**	**0.24**

Table 7: Accuracy for Qwen2.5 7B under two different input formats (A and B) with structured and unstructured inputs, with and without intermediate reasoning steps.



(a) Average attention weights across layers for Llama3.2 8B. We report attention from selected output tokens to both input and output tokens. Attention to the key partition-related tokens peaks consistently between layers 13 and 18, motivating our choice of this interval for full attention visualization.



(b) Average attention weights across layers for Gemma3-4B. We report attention from selected output tokens to both input and output tokens (as defined in Figure 4a). Attention magnitude peaks between layers 21 and 23.

Figure 12: Layer-wise attention analysis for Llama3.2-8B and Gemma3-4B. For each model, we visualize the average attention from selected output tokens to salient input tokens (notably the final item and comma of the partition). These trends are used to identify the layer ranges with the strongest partition-relevant attention.

provided below:

```
Answer the question with just a number only
    (We've separated each group of items with
    "|" so you can calculate the final count
    easier).
Question: How many fruits are there in the
    following sentence?
 apple, apple, apple | apple, apple, apple,
    apple, apple | apple, apple, apple, apple
```

To mitigate token-specific artifacts, we performed all experiments using a set of distinct items. (see Appendix A.1)

Information Pathway Localization To localize the specific attention heads and layers mediating the flow of count information, we performed attention knockout experiments as outlined in Section 5.3. This analysis utilizes a prompt that elicits specific System-2 behavior by explicitly separating partition counting from final aggregation. We employed a strict instructional format forcing the model to output intermediate counts before the final total. This allows us to measure the causal impact of blocking specific attention heads on the accuracy of both intermediate transfer and final summation (the exact prompt template is provided below).

```
I will provide an input text containing fruits
    separated by the '|' delimiter. Your goal
    is to count the items in each section and
    provide a summation.

Input Text:
 apple, apple, apple | apple, apple, apple,
    apple, apple | apple, apple, apple, apple
```

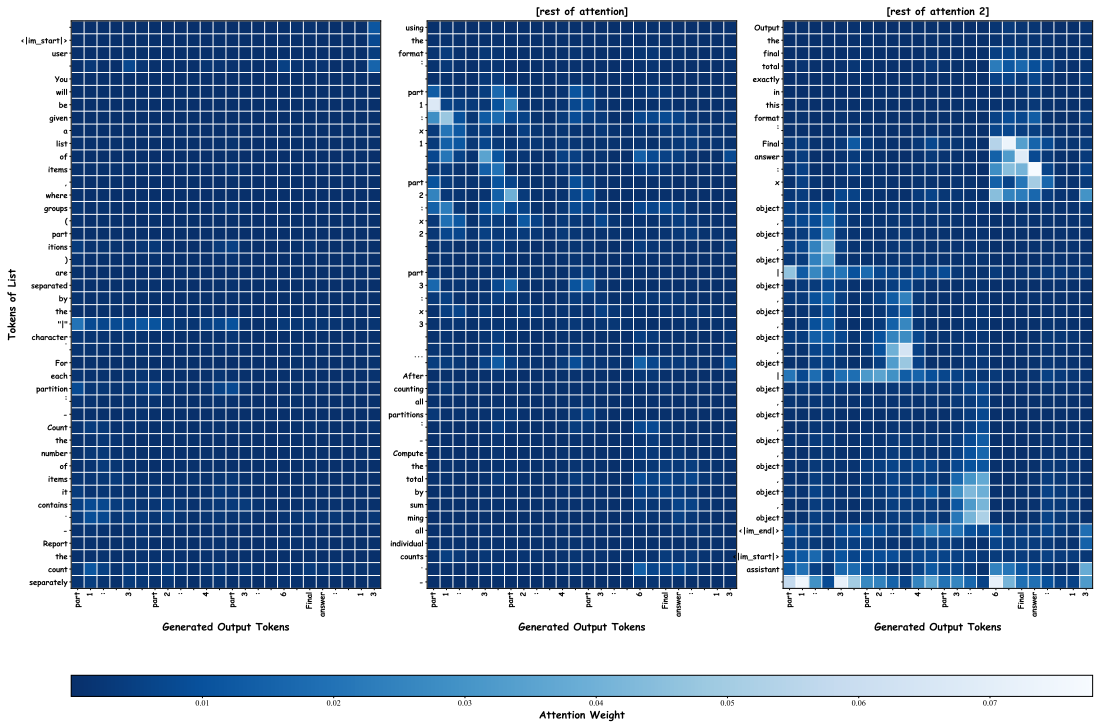


Figure 13: Full attention map for Qwen2.5-7B. The x-axis corresponds to output tokens and the y-axis to input tokens. The token immediately preceding the generated number of parts exhibits the strongest attention toward the final item and trailing comma of each partition matches its own segment.

```
Task:
1. Identify each partition separated by '|'.
2. Count the number of fruits in each specific
    partition.
3. Sum the counts of all partitions.

You must strictly follow this format, adapting
    the number of parts to the actual input:
part1: <count1>, part2: <count2>, ... , final
    count: <total_count>
```

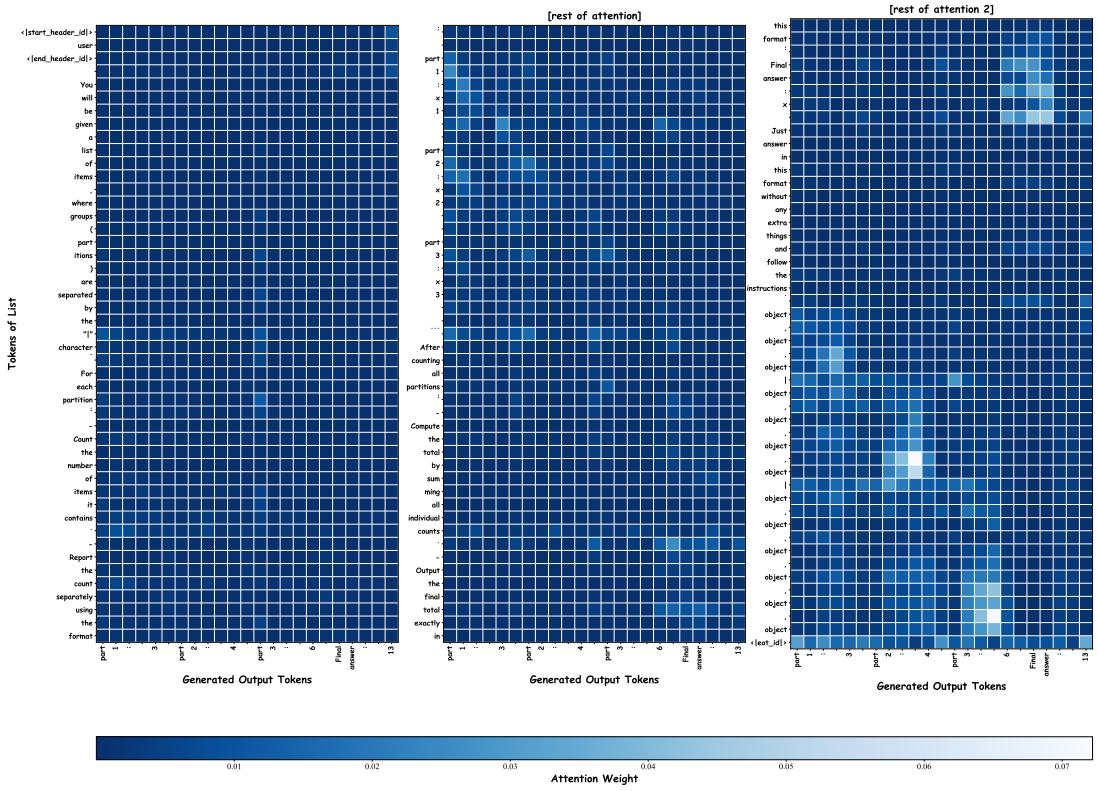


Figure 14: Full attention map for Llama3.2-8B. The x-axis corresponds to output tokens and the y-axis to input tokens. The token immediately preceding the generated number of parts exhibits the strongest attention toward the final item and trailing comma of each partition matches its own segment.

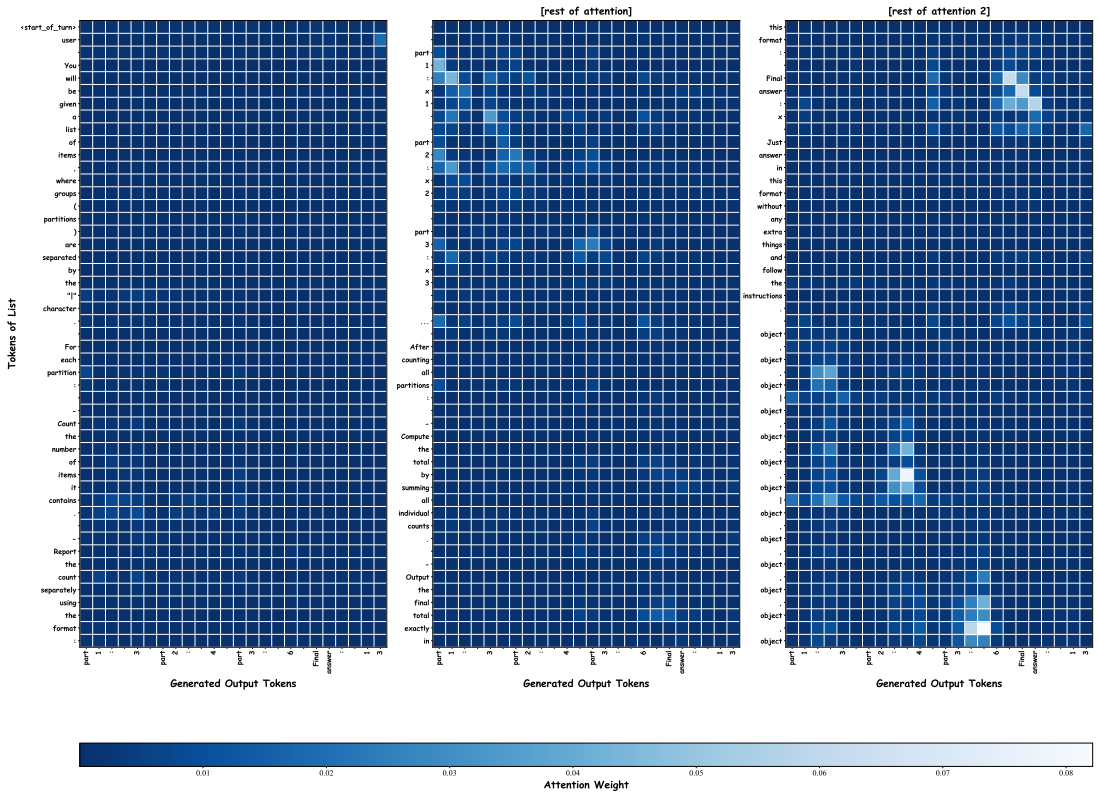


Figure 15: Full attention map for Gemma3-4B. The x-axis corresponds to output tokens and the y-axis to input tokens. The token immediately preceding the generated partition-level count exhibits the strongest attention toward the final item and trailing comma of each partition matches its own segment.