

# 000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 ATTENTION LOCALIZATION THROUGH SEPARATOR TOKENS: UNLOCKING LONG NUMERICAL SEQUENCE PROCESSING IN LLMs

Anonymous authors

Paper under double-blind review

## ABSTRACT

Despite possessing massive context windows, Large Language Models (LLMs) exhibit a sharp decline in performance when processing long numerical sequences, a critical failure for precision-sensitive applications. We identify the root cause as the models’ inability to focus attention on a manageable sequence segment, leading to dispersed attention and inaccurate results. To address this, we introduce **Separate Numerical Sequences** (SepNS), a training-free inference framework that guides LLMs by strategically inserting separators into numerical inputs. This simple modification encourages a “separate and focus” strategy, which we verify through attention analysis showing that separators induce localized focus on distinct segments. Extensive experiments on nine high-performance LLMs show SepNS substantially boosts accuracy, achieving average gains of **35.6%** across all evaluated datasets with less overhead. Our work demonstrates that simple, structured input formatting acts as a powerful attention-focusing mechanism, unlocking long numerical processing capabilities in LLMs without any retraining.

## 1 INTRODUCTION

Recent advances in Large Language Models (LLMs) have dramatically expanded their contextual capacity, with some models now supporting context windows of up to millions of tokens (Anthropic, 2025; Gemini-Team, 2025; Meta-AI, 2025; Team et al., 2025a;b), enabling processing of increasingly complex numerical data in domains such as weather forecasting (Gao et al., 2025) and stock analysis (Huang et al., 2021). However, a large context window does not guarantee that LLMs can analyze long sequences flawlessly (Hosseini et al., 2025). Empirical evidence suggests that LLMs experience substantial performance degradation when processing inputs that exceed 10-20% of their maximum context length (Kuratov et al., 2024; Liu et al., 2024b; Li et al., 2025c). Even in basic tasks like counting 1’s in the numerical sequences, LLMs’ performance declines as the length increases. As Figure 1(A) shows, accuracy drops by up to 70% as sequences grow from 2–32 to 256–512 elements in 6 fundamental tasks with the vanilla method. These fundamental deficiencies severely limit the deployment of LLMs in precision-critical scenarios, where numerical errors can propagate and lead to system failures, highlighting the urgent need to enhance LLMs’ long sequence processing capabilities for reliable data-intensive applications.

Existing approaches to enhance long-context processing can be broadly categorized into three paradigms. First, attention mechanisms have successfully extended context windows and accelerated inference (Leviathan et al., 2025; Lai et al., 2025; Liu et al., 2024a), but they fundamentally fail to resolve precision issues inherent in numerical sequence processing. Second, while content processing strategies such as summarization (Hosseini et al., 2025; Liu et al., 2025; Li et al., 2025b) and reordering (Peysakhovich & Lerer, 2023; Chen et al., 2024b) have shown promise for textual content, they are inherently incompatible with numerical sequences where order and completeness are mathematically critical. Third, specialized tokenization approaches (Yang et al., 2025b) require extensive retraining, which incurs prohibitive computational costs and risks degrading the model’s general-purpose capabilities. These limitations motivate our central research questions:

*What fundamental factors limit LLMs’ ability to process long numerical sequences? How can we enhance this capability without additional training?*

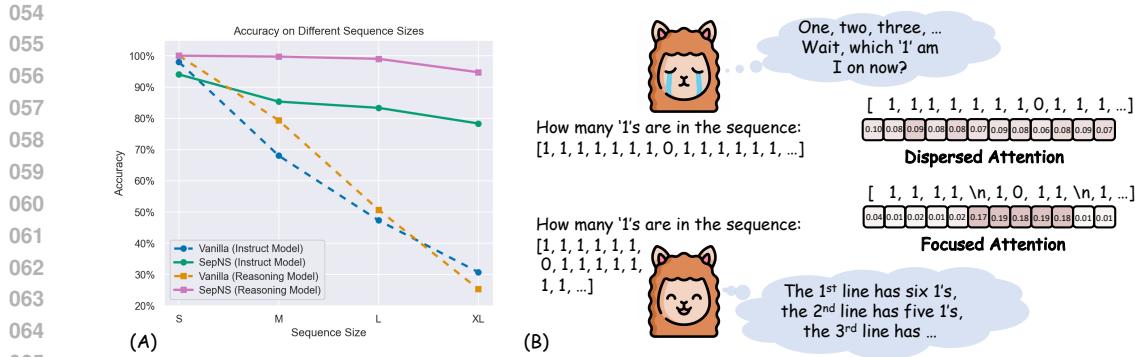


Figure 1: (A) Average accuracy across six synthetic tasks, performance drops sharply with increasing numerical sequence length (S: 2–32, M: 33–128, L: 129–256, XL: 257–512), while SepNS remains largely unaffected. (B) LLMs struggle with long numerical sequences due to dispersed attention, whereas SepNS uses separator tokens to maintain focused attention.

To answer these questions, we investigate the attention map for LLMs’ sequence processing and analyze the underlying mechanisms behind their performance degradation. Our systematic analysis reveals a fundamental insight: LLMs’ performance degradation on long numerical sequences stems from their limited capacity for focused attention on a separated sequence segment. As demonstrated in Figure 1(B), when confronted with long numerical sequence tasks, LLMs struggle to add natural breakpoints like humans do and process information in manageable chunks, leading to dispersed attention throughout the entire sequence. Based on this insight, we propose a simple yet effective approach to guide LLMs toward solving problems by introducing separators that partition long sequences into shorter, more manageable segments.

Building on these insights, we propose **SepNS**, a training-free inference framework that enhances LLMs’ long numerical sequence processing through strategic separator insertion. SepNS addresses the limitations through two key methodological contributions. First, we introduce systematic separator insertion, where separators are strategically placed within numerical sequences to create segmentation boundaries. This simple yet effective modification transforms intractable long sequences into multiple manageable segments that align with LLMs’ reliable processing capacity, directly mitigating the precision degradation issue observed in Figure 1(A). Second, we conduct a comprehensive analysis of attention patterns to reveal the underlying mechanisms through both theoretical analysis and experimental validation. Our investigation shows that separators cause specific attention heads to focus predominantly on local sequence segments rather than dispersing attention across the entire sequence. This localized attention pattern enables more precise numerical processing while preserving global context by integrating information across segments.

We conduct comprehensive experiments across 9 high-performance LLMs, evaluating performance on six synthetic tasks and four real numerical sequence processing domains. Our results demonstrate that SepNS substantially outperforms both chain-of-thought (Wei et al., 2022) and one-shot (Yu et al., 2022) prompting strategies, achieving significant average accuracy gains of 35.6% across all evaluated datasets. Notably, these performance gains are achieved with reduced computational overhead: the method requires no additional training and actually reduces the inference burden compared to baseline approaches.

This work makes three key contributions to enhancing LLMs’ capabilities for processing long numerical sequences. First, we systematically identify and characterize the fundamental bottleneck limiting LLMs’ performance on long numerical sequences: dispersed attention to the entire numerical sequence. Second, we introduce SepNS, a training-free framework that mitigates this issue by strategically inserting separators into the input. Third, we provide comprehensive theoretical guarantees and empirical validation across diverse datasets, establishing the effectiveness, efficiency, and interpretability of our approach. Our findings demonstrate that simple input format changes can unlock substantial performance gains in LLMs’ long numerical sequence processing.

108 

## 2 RELATED WORK

110 **Progress and Challenges in Long-Text Processing.** Although LLMs now support millions of  
 111 tokens (Anthropic, 2025; Gemini-Team, 2025; Team et al., 2025b), they still face significant chal-  
 112 lenges in long-context processing. Liu et al. (2024b) discover the “Lost in the Middle” phenomenon,  
 113 where LLMs ignore key information in the middle sections of long documents. To address this, re-  
 114 searchers propose solutions like MiniCache (Liu et al., 2024a) for efficient KV cache compression.  
 115 And Hosseini et al. (2025) emphasizes that simply increasing context windows cannot guarantee  
 116 perfect long sequence analysis.

117 **Deficiencies in Numerical Understanding.** Unlike semantic tasks, numerical tasks require un-  
 118 derstanding numbers as continuous quantities rather than discrete symbols, creating a fundamental  
 119 mismatch with LLMs’ token-based processing architecture. Research reveals that LLMs struggle  
 120 with maintaining numerical precision in long sequences, often exhibiting digit transposition errors  
 121 and magnitude confusion (Li et al., 2025a). Studies on mathematical reasoning demonstrate that  
 122 LLMs struggle with multi-step numerical calculations (Wei et al., 2022). Tokenization approaches  
 123 specifically designed for numerical data require extensive retraining, which is associated with pro-  
 124hibitive costs (Kudo & Richardson, 2018). Furthermore, research on numerical robustness reveals  
 125 that models often fail to maintain precision in long numerical sequences (Hendrycks et al., 2021),  
 126 highlighting the need for specialized approaches to enhance numerical understanding capabilities.

127 **Limitations of Tool Calling.** While tool calling capabilities of LLMs have rapidly developed  
 128 through frameworks like ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023), they struggle  
 129 with tasks requiring multi-source information integration. Studies show that even with access to ex-  
 130 ternal APIs and code execution environments (Gao et al., 2023), models face challenges in complex  
 131 reasoning scenarios that demand synthesis of heterogeneous information sources (Press et al., 2022).  
 132 Additionally, models exhibit reliability issues in long sequence processing, as reported by Welleck  
 133 et al. (2019), who noted systematic failures in maintaining sequence fidelity. Advanced models  
 134 also show unexpected character insertions during numerical sequence tasks, indicating persistent  
 135 challenges in precise sequence tasks (Zhang et al., 2022).

136 

## 3 METHOD

139 In this section, we begin by formally defining the problem of processing a long numerical sequence.  
 140 Through preliminary experiments, we then demonstrate that LLMs exhibit significant limitations in  
 141 accurately repeating such sequences. To address this critical issue, we propose **Separate Numerical**  
 142 **Sequences** (SepNS), a training-free and plug-and-play method designed to enhance LLMs’ per-  
 143 formance by strategically separating the input sequences. Finally, we provide a theoretical explanation  
 144 for the effectiveness of our proposed method.

145 

### 3.1 PROBLEM DEFINITION

146 We define a class of numerical sequence reasoning problems in which, given a numerical sequence  
 147  $s = \{a_1, a_2, \dots, a_n\}$  of length  $|s| = n$ , the task is to answer a natural language query  $q$  based on this  
 148 sequence. This class of problems is characterized by three core properties:

149 **Completeness Dependency.** Correct problem solving requires complete and accurate access to the  
 150 entire numerical sequence. Any absence, modification, or omission of any element  $a_i$  may lead to  
 151 deviations or errors in the final answer.

152 **Natural Language Understanding Requirement.** Questions are posed in natural language, which  
 153 requires accurate comprehension of user intent, including implicit conditional constraints, temporal  
 154 scope limitations, and domain-specific semantic meanings.

155 **Composite Reasoning Complexity.** The tasks exhibit high complexity due to the need for inte-  
 156 grating natural language understanding with numerical computation, involving conditional filtering,  
 157 sequential pattern recognition, and multi-step logical reasoning.

158 For instance, given a stock trading data sequence, the question “*Excluding non-trading days, how*  
 159 *many times did the open price of stock rise for three or more consecutive days?*” requires the

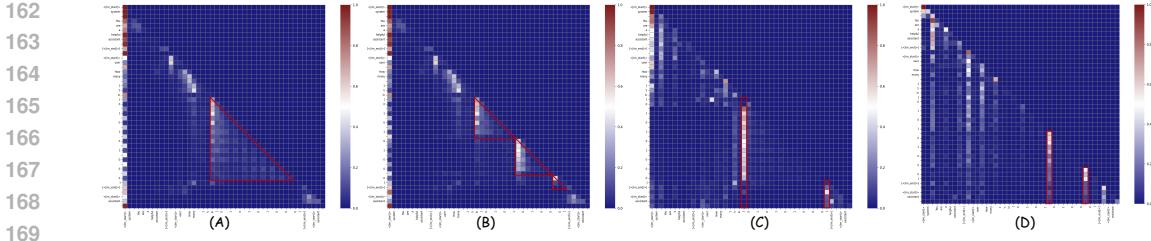


Figure 2: (A) and (C) visualize the attention scores given the input: “... How many 1’s in [0 1 0 1 0 1 0 0 0] ...”. (B) and (D) show the attention scores for the input with segmentation: “... How many 1’s in [0 1 0 1 \n 0 1 0 0 \n 0] ...”. (A) exhibits dispersed attention across the entire sequence. (B) demonstrates segment-focused attention. (C) highlights heightened attention at the beginning and end of the numerical sequence. (D) shows increased attention allocation to the separator token.

model to simultaneously understand the constraint condition of “excluding non-trading days”, the sequential pattern of consecutive rises, and the counting requirement of three or more days.

### 3.2 THE REPETITION DILEMMA

Existing research (Dong et al., 2025; Pimentel et al., 2025; Junchi Yao, 2025) has shown that LLMs are prone to significant repetition errors. To quantify this limitation in the context of numerical sequences, we conduct preliminary experiments requiring LLMs to reproduce numerical sequences word-for-word. Our findings reveal systematic performance degradation as sequence length increases (see Appendix B for details), with a particularly steep decline observed beyond a critical threshold. Specifically, when sequences exceed 256 floating-point numbers, only 14% of attempts successfully repeat the sequence. This dramatic performance decline reveals the root cause of the long numerical sequence processing limitation, which stems from the LLM’s limited capacity for understanding long numerical sequences, leading to frequent failures in preserving sequence integrity. This inherent deficiency ultimately results in the inability to construct accurate function calls or answer queries correctly, regardless of the availability of external tools.

### 3.3 ATTENTION LOCALIZATION THROUGH SEPARATOR TOKENS

To understand the underlying mechanisms behind these failures, we analyze attention patterns during the processing of long numerical sequences. As shown in Figure 2(A), we observe that LLMs tend to distribute attention across the entire sequence when processing a long numerical sequence. This behavior differs significantly from the “divide-and-conquer” strategy employed by humans when processing long sequences. Humans typically partition long sequences into several segments and process them sequentially with focused attention. Figure 2(C) reveals an interesting finding that while LLMs distribute attention across the entire sequence, attention weights tend to concentrate at the beginning and end of sequences. This finding aligns with research from Chen et al. (2024a), which demonstrates that LLMs exhibit attention concentration on certain special tokens (e.g., start/end markers in sequences, punctuation marks in sentences, and other separators); furthermore, the semantic embedding vectors of these separators often encapsulate key information from their preceding segments. Based on these observations, we propose SepNS that guides LLMs to focus attention on local segments rather than the global sequence by artificially introducing specific separators into sequences.

Formally, given a numerical sequence  $s$  of length  $n$ , SepNS transforms it into a structured format by periodically inserting separators. We define the transformation function as follows:

$$\text{SepNS}(s, k) = \{a_1, a_2, \dots, a_k\} \oplus \text{sep} \oplus \{a_{k+1}, \dots, a_{2k}\} \oplus \text{sep} \oplus \dots \oplus \{a_{n-r+1}, \dots, a_n\}, \quad (1)$$

where  $k$  denotes the segment size,  $\text{sep}$  represents the separator token (e.g., “\n”),  $\oplus$  denotes the concatenation operation, and  $r$  is the remainder of  $n$  divided by  $k$ . After introducing separators into sequences, we observe that transformer models exhibit a distinctive attention pattern: certain attention heads ignore contextual information before separators, and focus attention on the current separator and the numerical sequence following it, as illustrated in Figure 2(B, D).

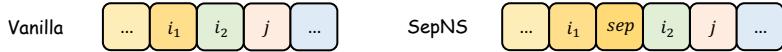
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269

Figure 3: The separator token  $sep$  in SepNS summarizes the content up to the current position. It exhibits a high score  $\mathbf{Q}_{i_1} \cdot \mathbf{K}_{sep}^T$  for token  $i_1$  (within its segment), but a low score  $\mathbf{Q}_{i_2} \cdot \mathbf{K}_{sep}^T$  for token  $i_2$  outside the summarized segment.

### 3.4 THEORETICAL EXPLANATION

We analyze how separator tokens mechanistically alter attention computation between tokens in different segments. For vanilla attention without separators, given sequence  $s = \{a_1, a_2, \dots, a_n\}$ , the attention weight between positions  $i$  and  $j$  is computed using query vector  $\mathbf{Q}_i$ , key vector  $\mathbf{K}_j$ , and key dimension  $d_k$ :

$$A_{\text{vanilla}}[i, j] = \frac{\exp(\mathbf{Q}_i \cdot \mathbf{K}_j^T / \sqrt{d_k})}{\sum_{l=1}^n \exp(\mathbf{Q}_i \cdot \mathbf{K}_l^T / \sqrt{d_k})}. \quad (2)$$

With SepNS, the sequence becomes  $s' = \{a_1, \dots, a_k, sep, a_{k+1}, \dots, a_{2k}, sep, \dots\}$  with length  $n'$ . The attention weight between positions  $i$  and  $j$  is:

$$A_{\text{SepNS}}[i, j] = \frac{\exp(\mathbf{Q}_i \cdot \mathbf{K}_j^T / \sqrt{d_k})}{\sum_{l=1}^{n'} \exp(\mathbf{Q}_i \cdot \mathbf{K}_l^T / \sqrt{d_k})}. \quad (3)$$

Denote  $S_{sep}$  as the set of all tokens before the current separator  $sep$ . As shown in Chen et al. (2024a), separator tokens  $sep$  summarize tokens up to the current position, thus  $sep$  exhibit high attention scores  $\mathbf{Q}_{i_1} \cdot \mathbf{K}_{sep}^T$  for tokens  $i_1 \in S_{sep}$  but low scores  $\mathbf{Q}_{i_2} \cdot \mathbf{K}_{sep}^T$  for token  $i_2 \notin S_{sep}$  (Figure 3). When computing attention from position  $i_1 \in S_{sep}$  to position  $j \notin S_{sep}$ , separator tokens dramatically increase the denominator of  $A_{\text{SepNS}}[i, j]$  through terms  $\sum_{i_1 \in S_{sep}} \exp(\mathbf{Q}_{i_1} \cdot \mathbf{K}_{sep}^T / \sqrt{d_k})$ .

Thus for token  $i_1 \in S_{sep}$  and  $j \notin S_{sep}$ , the cross-segment attention  $A_{\text{SepNS}}[i_1, j]$  becomes significantly suppressed as:

$$\frac{A_{\text{SepNS}}[i_1, j]}{A_{\text{vanilla}}[i_1, j]} = \frac{\sum_{l=1}^n \exp(\mathbf{Q}_{i_1} \cdot \mathbf{K}_l^T / \sqrt{d_k})}{\sum_{l=1}^{n'} \exp(\mathbf{Q}_{i_1} \cdot \mathbf{K}_l^T / \sqrt{d_k})} \ll 1. \quad (4)$$

In contrast, for tokens  $i_2 \notin S_{sep}$ ,  $A_{\text{SepNS}}[i_2, j]/A_{\text{vanilla}}[i_2, j] \approx 1$  since  $\exp(\mathbf{Q}_{i_2} \cdot \mathbf{K}_{sep}^T / \sqrt{d_k})$  is small, maintaining high attention to tokens in the current segment.

This mathematical analysis reveals the underlying mechanism: separator tokens act as “attention sinks” that absorb attention weight otherwise dispersed to cross-segment positions. The high query-key similarity between tokens (e.g.,  $i_1$ ) within the summarized segment and separator tokens (e.g.,  $sep$ ) effectively “shields” these positions from attending to distant segments (e.g., segment with  $i_2, j$ ), thereby localizing attention within current segments and creating structured attention boundaries without explicit masking. See Appendix F for a detailed proof.

## 4 EXPERIMENTS

To evaluate our approach, we structure experiments around the following research questions (RQs), examining performance gains and robustness.

**RQ1 – Effectiveness.** Does our proposed method consistently enhance model performance across diverse tasks and model architectures, demonstrating its general applicability?

**RQ2 – Robustness.** Do any factors modulate the effectiveness of our method, and what actionable insights can we provide for optimal deployment in different scenarios?

In this section, we present a comprehensive experimental evaluation designed to systematically address these research questions. We begin by detailing our experimental setup, including the carefully curated datasets, evaluation metrics, selected LLMs, and baseline methods. We then present our experimental results and analyses across multiple dimensions to address the above research questions, demonstrating the superior performance of our method.

270 4.1 EXPERIMENTAL SETTINGS  
271272 4.1.1 DATASET  
273274 We design two datasets to conduct an in-depth investigation of LLMs’ capabilities for processing  
275 long numerical sequences: a synthetic dataset  $\mathcal{D}_{\text{syn}}$  and a real dataset  $\mathcal{D}_{\text{real}}$ .276 For  $\mathcal{D}_{\text{syn}}$ , we construct sequences of varying lengths comprising both integer and floating-point  
277 numbers. We categorize these sequences into four length intervals: S (short) for sequences con-  
278 taining [2, 32] numbers, M (medium) for (32, 128] numbers, L (large) for (128, 256] numbers, and  
279 XL (extra-large) for (256, 512] numbers. We formulate six distinct task types: (1) *max-int*, which  
280 requires identifying the index of the maximum integer in an integer sequence; (2) *min-int*, which  
281 locates the index of the minimum integer; (3) *max-float* and (4) *min-float*, which perform analogous  
282 operations on floating-point sequences; (5) *indexing*, which determines the position of the last oc-  
283 currence of 1 in a binary sequence; and (6) *counting*, which counts the total number of 1s in a binary  
284 sequence. Each task comprises 200 samples, with 50 samples distributed across each of the four  
285 length categories.286 Building upon the prior work of [Li et al. \(2025a\)](#), we construct  $\mathcal{D}_{\text{real}}$  to assess model performance on  
287 practical numerical reasoning tasks. This dataset comprises four distinct categories, each containing  
288 200 samples. The categories include: (1) *number-string*, which involves counting numerals in al-  
289 phanumeric sequences; (2) *number-list*, requiring logical reasoning over numerical sequences; and  
290 (3-4) *stock* and *weather*, both constructed from real-world datasets with human-generated questions.  
291 Notably, any failure to process a single value in these tasks inevitably results in an incorrect final  
292 answer, making them particularly challenging. See [Appendix C](#) for details and examples.293 4.1.2 EVALUATION METRICS  
294295 To comprehensively evaluate our proposed method, we assess both performance and robustness  
296 using the following metrics:297 **Accuracy (Acc).** This is our primary performance metric, measuring the percentage of correctly  
298 answered questions. Let  $N_{\text{correct}}$  denote the number of responses that answer correctly out of  $N$  total  
299 questions. Accuracy is computed as:

300 
$$\text{Accuracy} = N_{\text{correct}}/N. \quad (5)$$
  
301

302 **Answer Rate (AR).** This metric captures the model’s capability to generate valid responses across  
303 all test instances. We observe that models may occasionally fail to produce meaningful output,  
304 resulting in null responses. AR quantifies the proportion of questions for which the model generates  
305 a valid, non-null response. Let  $N_{\text{valid}}$  denote the number of valid responses out of  $N$  total questions:

306 
$$\text{Answer Rate} = N_{\text{valid}}/N. \quad (6)$$
  
307

308 4.1.3 BASE MODELS  
309310 We conduct a comprehensive evaluation across 9 LLMs, representing diverse architectures, par-  
311 ameter scales, and training paradigms from both open-source and proprietary domains. **Open-source**  
312 **models:** Our selection includes the Qwen3 family ([Yang et al., 2025a](#)), spanning 0.6B to 30B pa-  
313 rameters with both dense and Mixture-of-Experts architectures ([Fedus et al., 2022](#); [Zhou et al.,](#)  
314 [2023](#)), available in instruct and reasoning modes, alongside the QwQ-32B model. We also eval-  
315 uate the DeepSeek series ([Guo et al., 2025](#)), including the recent DeepSeek-R1 and DeepSeek-V3  
316 variants, which are known for their strong reasoning capabilities. **Proprietary models:** We assess  
317 Claude-3.7-Sonnet from Anthropic ([Anthropic, 2025](#)). Additionally, we evaluate Google’s Gemini-  
318 2.5-Pro ([Gemini-Team, 2025](#)), which showcases multimodal understanding capabilities, and two  
319 variants from OpenAI’s GPT-4 series ([Achiam et al., 2023](#)): GPT-4.1 and GPT-4o. See [Section 7](#)  
320 for a detailed version of the models.321 4.1.4 BASELINES  
322323 To establish comprehensive benchmarks for evaluating our proposed method, we employ two base-  
line approaches except the vanilla method. **Chain-of-Thought:** In the chain-of-thought setting ([Wei](#)

324  
 325 Table 1: The average answer rate and accuracy of each task among the four methods. “Incr.” indi-  
 326 cates the percentage improvement of SepNS over Vanilla. **Green** and **red** indicate improvement and  
 327 degradation in performance metrics, respectively.

Task	Answer Rate ( $\uparrow$ )				Accuracy ( $\uparrow$ )			
	Vanilla	CoT	One-shot	SepNS (Incr.)	Vanilla	CoT	One-shot	SepNS (Incr.)
counting	100.0%	100.0%	100.0%	100.0% (+0.0%)	42.7%	41.2%	37.9%	76.7% (+79.6%)
indexing	100.0%	100.0%	100.0%	100.0% (+0.0%)	38.8%	33.4%	34.9%	86.6% (+123.0%)
max-float	100.0%	100.0%	100.0%	100.0% (+0.0%)	63.9%	60.5%	63.3%	81.0% (+26.8%)
max-int	100.0%	100.0%	100.0%	100.0% (+0.0%)	75.3%	68.8%	71.0%	92.4% (+22.7%)
min-float	100.0%	100.0%	100.0%	100.0% (+0.0%)	63.1%	60.2%	60.4%	79.3% (+25.7%)
min-int	100.0%	100.0%	100.0%	100.0% (+0.0%)	74.3%	68.1%	66.5%	93.0% (+25.1%)
number-string	96.6%	96.3%	96.3%	99.0% (+2.5%)	81.6%	81.0%	81.7%	81.7% (+0.1%)
number-list	78.2%	77.6%	73.4%	79.1% (+1.1%)	36.3%	35.4%	32.6%	36.7% (+0.9%)
stock	64.1%	63.1%	63.3%	73.6% (+14.7%)	13.2%	13.7%	13.4%	27.4% (+107.6%)
weather	66.0%	65.6%	65.9%	76.8% (+16.3%)	26.7%	27.2%	27.8%	44.6% (+67.1%)
Average	90.5%	90.3%	89.9%	92.8% (+2.6%)	51.6%	49.0%	48.9%	69.9% (+35.6%)

338  
 339  
 340 et al., 2023), the model is guided by prompt instructions to complete the task step by step. **One-  
 341 shot Learning:** For the one-shot baseline (Yu et al., 2022), we provide the model with a single  
 342 demonstration example that illustrates the desired input-output mapping for the target task.

## 343 4.2 EXPERIMENTAL RESULTS

344 In this section, we present a comprehensive evaluation designed to systematically address our re-  
 345 search questions. Each research question is analyzed and substantiated through multiple comple-  
 346 mentary perspectives, providing thorough empirical evidence for our claims.

### 347 4.2.1 EFFECTIVENESS (RQ1)

348 **Across tasks.** We evaluate our method against baselines on six synthetic and four real tasks, report-  
 349 ing Accuracy and Answer Rate (additional metrics (e.g., “Response Length”) in Appendix D). As  
 350 shown in Table 1, our analysis reveals critical deficiencies in standard LLMs when processing long  
 351 numerical sequences. Notably, popular enhancement methods, such as Chain-of-Thought (CoT)  
 352 and One-shot prompting, fail to address this fundamental problem. In fact, they prove detri-  
 353 mental, with average accuracies dropping from 51.6% (Vanilla) to 49.0% (CoT) and 48.9% (One-shot).  
 354 This strongly suggests that LLM’s failures stem not from an insufficient reasoning ability but from  
 355 a fundamental inability to properly parse and manage numerical sequences.

356 In contrast, our SepNS framework demonstrates remarkable efficacy, substantially elevating perfor-  
 357 mance across all tasks. By strategically structuring input, SepNS boosts average accuracy to 69.9%,  
 358 a significant 35.6% relative improvement over the vanilla baseline. This improvement extends be-  
 359 yond synthetic data—on real tasks, complex numerical tasks, SepNS proves equally effective. Ac-  
 360 curacy on stock and weather datasets increases dramatically by +107.6% and +67.1%, respectively.  
 361 SepNS also enhances reliability, increasing average answer rate to 92.8% (+2.6%), ensuring models  
 362 provide both more accurate and consistent responses. These results confirm that SepNS effectively  
 363 rectifies core weaknesses in LLM sequence understanding, enabling significant improvements in  
 364 numerical processing without requiring model modifications.

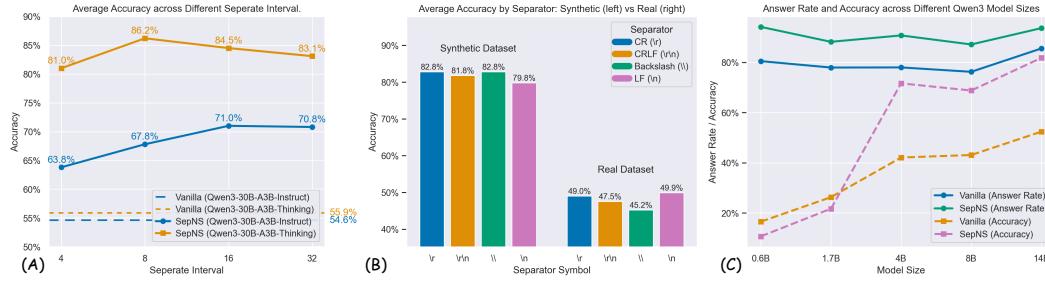
365 **Across models.** We evaluate 9 diverse high-performance LLMs and summarize model-wise per-  
 366 formance gains over vanilla baselines. Results are presented in model-wise tables to validate the  
 367 general applicability of our approach across different model architectures and scales.

368 The results in Table 2 demonstrate that SepNS provides consistent performance enhancements across  
 369 all tested architectures, achieving a remarkable +35.6% average accuracy boost over the vanilla  
 370 baseline. The framework yields significant gains irrespective of model size or origin. Open-source  
 371 models like QwQ-32B and Qwen3-8B exhibit significant improvements of +69.0% and +53.0%,  
 372 respectively, suggesting SepNS effectively unlocks numerical processing capabilities. This trend  
 373 extends to advanced proprietary models—Claude-3.7-Sonnet (+38.0%), Gemini-2.5-Pro (+34.3%),  
 374 and GPT-4.1 (+35.6%), all of which derive significant benefits. These indicate that difficulty han-  
 375 dling long numerical sequences is a fundamental limitation inherent in current LLM architectures

378

379  
380  
381  
Table 2: The average answer rate and accuracy of each model among the four methods. “Incr.”  
indicates the percentage improvement of SepNS over Vanilla. **Green** and **red** indicate improvement  
and degradation in performance metrics, respectively.

Model	Answer Rate ( $\uparrow$ )				Accuracy ( $\uparrow$ )			
	Vanilla	CoT	One-shot	SepNS (Incr.)	Vanilla	CoT	One-shot	SepNS (Incr.)
Qwen3-8B	79.4%	79.5%	75.2%	87.3% (+9.9%)	45.5%	40.3%	38.5%	69.6% (+53.0%)
Qwen3-30B-A3B	80.6%	80.9%	81.0%	90.2% (+11.9%)	54.3%	52.7%	52.3%	80.7% (+48.6%)
QwQ-32B	72.7%	72.2%	72.7%	75.6% (+4.0%)	34.2%	28.9%	28.5%	57.8% (+69.0%)
DeepSeek-V3	100.0%	100.0%	100.0%	100.0% (+0.0%)	45.7%	44.5%	45.2%	50.9% (+11.4%)
DeepSeek-R1	99.9%	99.9%	99.8%	99.9% (+0.0%)	61.1%	56.6%	57.9%	70.5% (+15.4%)
Claude-3.7-Sonnet	99.8%	99.9%	100.0%	99.9% (+0.1%)	57.3%	57.6%	54.2%	79.1% (+38.0%)
Gemini-2.5-Pro	83.4%	81.8%	82.0%	84.6% (+1.4%)	58.9%	48.0%	56.9%	79.1% (+34.3%)
GPT-4.1	99.6%	99.6%	99.8%	99.5% (-0.1%)	61.0%	60.0%	60.0%	82.7% (+35.6%)
GPT-4o	99.0%	98.5%	98.6%	98.6% (-0.4%)	46.4%	51.5%	44.5%	59.1% (+27.4%)
Average	<b>90.5%</b>	90.3%	89.9%	<b>92.8%</b> (+2.6%)	<b>51.6%</b>	48.9%	48.7%	<b>68.9%</b> (+35.6%)

402  
403  
404  
Figure 4: (A): Average Accuracy across different separate intervals. (B): Average Accuracy across  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365<br

432  
 433 Table 3: Accuracy comparison between instruction (Qwen3-30B-A3B-Instruct) and reasoning  
 434 (Qwen3-30B-A3B-Thinking) models across different separate intervals (4/8/16/32). **Bold** and  
 435 underlined values denote the highest and second-highest scores, respectively.

Size	Accuracy on Instruct Model ( $\uparrow$ )					Accuracy on Reasoning Model ( $\uparrow$ )				
	vanilla	4	8	16	32	vanilla	4	8	16	32
S	<b>98.0%</b>	88.7%	91.0%	94.0%	<u>95.0%</u>	<b>100.0%</b>	<b>100.0%</b>	<u>99.7%</u>	<b>100.0%</b>	99.3%
M	68.0%	<u>81.7%</u>	81.0%	<b>85.3%</b>	81.0%	79.3%	99.6%	<b>100.0%</b>	<u>99.7%</u>	<u>99.7%</u>
L	47.3%	70.3%	78.7%	<u>83.3%</u>	<u>79.7%</u>	50.7%	98.0%	<b>99.3%</b>	<u>99.0%</u>	97.0%
XL	30.7%	65.3%	68.7%	<u>78.3%</u>	<b>80.7%</b>	25.3%	81.2%	<u>92.0%</u>	<b>94.7%</b>	91.7%
Avg.	61.0%	76.5%	79.8%	<b>85.2%</b>	<u>84.1%</u>	63.8%	94.7%	<u>97.7%</u>	<b>98.3%</b>	96.9%

436  
 437  
 438  
 439  
 440  
 441  
 442  
 443  
 444 Figure 4(B) reveals systematic performance differentiation across separator types varying with task  
 445 complexity. For basic tasks  $\mathcal{D}_{\text{syn}}$ , all separators showed modest accuracy variations (79.8%–82.8%),  
 446 with unconventional separators CR and Backslash achieving optimal accuracy (82.8% each), po-  
 447 tentially due to novelty requiring enhanced attention. However, inversion emerged in complex sce-  
 448 narios: while CR and Backslash excelled in basic tasks, LF demonstrated superior performance  
 449 (49.9% vs. Backslash’s 45.2%) in complex tasks. This reversal suggests fundamental processing  
 450 strategy shifts across complexity levels. These patterns evidence sophisticated cognitive resource  
 451 allocation (Sweller, 1988; Sweller et al., 2011) in LLMs. Under low cognitive load, models al-  
 452 locate additional resources to separator adaptation, where novel separators benefit from enhanced  
 453 attention. Under high cognitive load, models prioritize core semantic processing, favoring minimal-  
 454 overhead separators. LF’s superior complex performance reflects the prevalence of training data and  
 455 processing efficiency, enabling more resources for task-specific reasoning over format adaptation.

456 **Model size.** We evaluated Qwen3 models at various parameter scales (0.6B, 1.7B, 4B, 8B, and  
 457 14B). The results, shown in Figure 4(C), demonstrate that the effectiveness of SepNS is scale-  
 458 dependent. For smaller models (0.6B), SepNS underperforms the vanilla baseline, suggesting a  
 459 minimum capacity requirement for effective separator interpretation. A critical inflection point oc-  
 460 curs at the 4B parameter level, where SepNS begins to yield performance gains. Beyond this scale,  
 461 the performance gap widens substantially, reaching over 80% accuracy on the 14B models.

462 **Reasoning vs. Instruction models.** We contrast reasoning variants against the instruction model,  
 463 reporting paired differences to assess whether reasoning ability affects separator sensitivity. Table 3  
 464 reveals powerful effects when applying SepNS to stronger reasoning models.

465 With the vanilla method, both model types suffer severe degradation on long sequences. On extra-  
 466 long sequences, vanilla accuracy drops to 30.7% (instruction) and 25.3% (reasoning), highlighting  
 467 shared weaknesses. However, SepNS reveals significant gaps: instruction models reach 85.2% peak  
 468 accuracy (interval 16), while reasoning models achieve a near-perfect 98.3%. On large sequences,  
 469 reasoning accuracy jumps from 50.7% to 99.3%, versus instruction’s modest 47.3% to 83.3% im-  
 470 provement. Reasoning models exhibit greater interval robustness, maintaining near-optimal per-  
 471 formance across intervals 8–32, while instruction models are more hyperparameter-sensitive. This  
 472 indicates SepNS provides structural decomposition that reasoning models uniquely exploit, convert-  
 473 ing challenging problems into manageable sub-tasks for state-of-the-art accuracy.

## 474 5 CONCLUSION

475  
 476 In this work, we identify and address a fundamental limitation of LLMs: their inability to main-  
 477 tain focused attention when processing long numerical sequences, resulting in severe performance  
 478 degradation in precision-critical applications. We introduced SepNS, a training-free framework that  
 479 strategically inserts separators to partition long sequences into manageable segments. Through  
 480 comprehensive evaluation across 9 high-performance models on 10 tasks, SepNS achieves a substantial  
 481 35.6% average accuracy improvement without computational overhead or retraining. Our analy-  
 482 sis reveals that separators induce localized attention patterns, transforming dispersed attention into  
 483 focused segment processing while preserving global context. This demonstrates that simple input  
 484 formatting serves as a powerful attention-focusing mechanism, unlocking significant numerical pro-  
 485 cessing capabilities and providing practical solutions for precision-critical applications.

486 **6 ETHICS STATEMENT**  
 487

488 Our research adheres to the ICLR Code of Ethics and raises no ethical concerns. The proposed  
 489 SepNS framework is a training-free inference technique that modifies input formatting without alter-  
 490 ing model parameters or requiring additional data collection. Our experiments utilize publicly  
 491 available models and synthetic datasets, with no involvement of human subjects, collection of per-  
 492 sonal data, or privacy risks. The method enhances model accuracy in numerical processing tasks  
 493 without introducing harmful capabilities or creating potential for misuse. All experimental evalua-  
 494 tions were conducted using established benchmarks and standard evaluation protocols. The research  
 495 makes a positive contribution to the field by addressing fundamental limitations in LLM numeri-  
 496 cal processing capabilities, with potential benefits for applications that require precise numerical  
 497 computation.

498 **7 REPRODUCIBILITY STATEMENT**  
 499

500 To facilitate reproducibility of our results, we provide comprehensive documentation and resources  
 501 across multiple components of this work.

503 **Code and Data.** Our proposed method is thoroughly detailed in Section 3, including algorithmic  
 504 descriptions and implementation specifics. Complete source code and datasets are available through  
 505 the anonymous repository at <https://anonymous.4open.science/r/SepNS>, with the  
 506 accompanying README file providing step-by-step instructions for execution and reproduction  
 507 of experiments.

508 **Theorem.** Rigorous theoretical foundations are established in Appendix F, which contains detailed  
 509 mathematical proofs and derivations supporting our theoretical claims. These materials collectively  
 510 provide researchers with the necessary resources to validate and build upon our contributions.

511 **Selected Models.** We evaluate diverse high-performance models across different experimental set-  
 512 tings. For the **main evaluation (RQ1)**, we use 9 high-performance LLMs:

- 514 1. Qwen3-8B: <https://huggingface.co/Qwen/Qwen3-8B>
- 515 2. Qwen3-30B-A3B: <https://huggingface.co/Qwen/Qwen3-30B-A3B>
- 516 3. QwQ-32B: <https://huggingface.co/Qwen/QwQ-32B>
- 517 4. DeepSeek-V3: <https://huggingface.co/deepseek-ai/DeepSeek-V3-0324>
- 518 5. DeepSeek-R1: <https://huggingface.co/deepseek-ai/DeepSeek-R1-0528>
- 519 6. Claude-3.7-Sonnet: <https://openrouter.ai/anthropic/clause-3.7-sonnet>
- 520 7. Gemini-2.5-Pro: <https://openrouter.ai/google/gemini-2.5-pro>
- 521 8. GPT-4.1: <https://openrouter.ai/openai/gpt-4.1>
- 522 9. GPT-4o: <https://openrouter.ai/openai/gpt-4o-2024-08-06>

525 For robustness evaluation (RQ2), **separator interval analysis**, we compare instruction and reason-  
 526 ing variants:

- 528 1. Qwen3-30B-A3B-Instruct: <https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507>
- 529 2. Qwen3-30B-A3B-Thinking: <https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507>

533 For **separator symbol analysis**, we use Qwen3-30B-A3B-Instruct (<https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507>).

535 For **model size analysis**, we evaluate across different parameter scales:

- 537 1. Qwen3-0.6B: <https://huggingface.co/Qwen/Qwen3-0.6B>
- 538 2. Qwen3-1.7B: <https://huggingface.co/Qwen/Qwen3-1.7B>
- 539 3. Qwen3-4B: <https://huggingface.co/Qwen/Qwen3-4B>

540 4. Qwen3-8B: <https://huggingface.co/Qwen/Qwen3-8B>  
 541 5. Qwen3-14B: <https://huggingface.co/Qwen/Qwen3-14B>

543 **REFERENCES**

545 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-  
 546 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical  
 547 report. *arXiv preprint*, abs/2303.08774, 2023.

548 Anthropic. System card: Claude opus 4 claude sonnet 4. <https://www-cdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf>, 2025.

551 Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo  
 552 Li, Weiyang Liu, and Chao Huang. SepLLM: Accelerate large language models by compressing  
 553 one segment into one separator. Vienna, Austria, 2024a.

555 Yuhang Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and  
 556 Rui Yan. Fortify the shortest stave in attention: Enhancing context awareness of large language  
 557 models for effective tool use. In *Proceedings of the 62nd Annual Meeting of the Association  
 558 for Computational Linguistics*, pp. 11160–11174, Bangkok, Thailand, 2024b. Association for  
 559 Computational Linguistics.

560 Yihong Dong, Yuchen Liu, Xue Jiang, Bin Gu, Zhi Jin, and Ge Li. Rethinking repetition problems  
 561 of LLMs in code generation. In *Proceedings of the 63rd Annual Meeting of the Association for  
 562 Computational Linguistics*, pp. 965–985, Vienna, Austria, 2025.

563 William Fedus, Barret Zoph, and Noam Shazeer. Switch transformer: Scaling to trillion parameter  
 564 models with simple and efficient sparsity. In *Journal of Machine Learning Research*, volume 23,  
 565 pp. 1–39, 2022.

566 Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and  
 567 Graham Neubig. Pal: Program-aided language models. In *Proceedings of the 40th International  
 568 Conference on Machine Learning*, pp. 10764–10799, Honolulu, HI, 2023.

570 Yuan Gao, Hao Wu, Ruiqi Shu, Huanshuo Dong, Fan Xu, Rui Chen, Yibo Yan, Qingsong Wen,  
 571 Xuming Hu, Kun Wang, Jiahao Wu, Qing Li, Hui Xiong, and Xiaomeng Huang. Oneforecast: A  
 572 universal framework for global and regional weather forecasting. *arXiv preprint*, abs/2502.00338,  
 573 2025.

574 Gemini-Team. Gemini 2.5 pro model card. [https://storage.googleapis.com/model-  
 576 cards/documents/gemini-2.5-pro.pdf](https://storage.googleapis.com/model-<br/>
  575 cards/documents/gemini-2.5-pro.pdf), 2025.

577 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,  
 578 Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in  
 579 LLMs via reinforcement learning. *arXiv preprint*, abs/2501.12948, 2025.

580 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,  
 581 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Pro-  
 582 ceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, vol-  
 583 ume 1, New Orleans, LA, 2021.

585 Peyman Hosseini, Ignacio Castro, Iacopo Ghinassi, and Matthew Purver. Efficient solutions for  
 586 an intriguing failure of LLMs: Long context window does not mean LLMs can analyze long  
 587 sequences flawlessly. In *Proceedings of the 31st International Conference on Computational  
 588 Linguistics*, pp. 1880–1891, Abu Dhabi, United Arab Emirates, 2025.

590 Yuxuan Huang, Luiz Fernando Capretz, and Danny Ho. Machine learning for stock prediction based  
 591 on fundamental analysis. In *IEEE Symposium Series on Computational Intelligence*, pp. 1–10,  
 592 Orlando, FL, 2021. IEEE.

593 Jianhua Xu Lijie Hu Mengdi Li Di Wang Junchi Yao, Shu Yang. Understanding the repeat curse in  
 large language models from a feature perspective. *arXiv preprint*, abs/2504.14218, 2025.

594 Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword  
 595 tokenizer and detokenizer for neural text processing. *arXiv preprint*, abs/1808.06226, 2018.  
 596

597 Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Y. Sorokin,  
 598 and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-  
 599 haystack. In *Advances in Neural Information Processing Systems 38*, Vancouver, Canada, 2024.

600 Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse  
 601 attention mechanism for efficient long-sequence inference. In *Proceedings of the 13th Interna-*  
 602 *tional Conference on Learning Representations*, Singapore, 2025.

603

604 Yaniv Leviathan, Matan Kalman, and Yossi Matias. Selective attention improves transformer. In  
 605 *Proceedings of the 13th International Conference on Learning Representations*, Singapore, 2025.

606

607 Haoyang Li, Xuejia Chen, Zhanchao Xu, Darian Li, Nicole Hu, Fei Teng, Yiming Li, Luyu Qiu,  
 608 Chen Jason Zhang, Qing Li, and Lei Chen. Exposing numeracy gaps: A benchmark to evaluate  
 609 fundamental numerical abilities in large language models. In *Findings of the 63th Annual Meeting*  
 610 *f the Association for Computational Linguistics*, pp. 20004–20026, Vienna, Austria, 2025a.

611 Taiji Li, Hao Chen, Fei Yu, and Yin Zhang. HERA: improving long document summarization using  
 612 large language models with context packaging and reordering. *arXiv preprint*, abs/2502.00448,  
 613 2025b.

614

615 Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhui Chen. Long-context LLMs struggle with  
 616 long in-context learning. *Transactions on Machine Learning Research*, 2025, 2025c.

617

618 Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Reza Haffari, and Bohan Zhuang. Minicache: Kv cache  
 619 compression in depth dimension for large language models. In *Advances in Neural Information*  
 620 *Processing Systems 38*, Vancouver, Canada, 2024a.

621

622 Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and  
 623 Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the*  
 624 *Association for Computational Linguistics*, 12:157–173, 2024b.

625

626 Ran Liu, Xian-Ling Mao, and Heyan Huang. Dscisum: Detailed summarization of long scientific  
 627 documents. *Knowledge-Based Systems*, 317:113409, 2025.

628

629 Meta-AI. The Llama 4 herd: The beginning of a new era of natively multimodal ai innovation.  
 https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

630

631 Alexander Peysakhovich and Adam Lerer. Attention sorting combats recency bias in long context  
 632 language models. *arXiv preprint*, abs/2310.01427, 2023.

633

634 Tiago Pimentel et al. Repetitions are not all alike: Distinct mechanisms sustain repetition in language  
 635 models. *arXiv preprint*, abs/2504.01100, 2025.

636

637 Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring  
 638 and narrowing the compositionality gap in language models. In *Findings of the 2022 Conference*  
 639 *on Empirical Methods in Natural Language Processing*, pp. 5687–5711, Abu Dhabi, United Arab  
 Emirates, 2022.

640

641 Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer,  
 642 Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to  
 643 use tools. *arXiv preprint*, abs/2302.04761, 2023.

644

645 John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:  
 257–285, 1988.

646

647 John Sweller, Paul Ayres, and Slava Kalyuga. *Cognitive Load Theory*. Springer, New York, 2011.  
 ISBN 978-1-4419-8125-7.

648 GLM-4.5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie,  
 649 Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu,  
 650 Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin  
 651 Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen,  
 652 Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi  
 653 Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin,  
 654 Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo  
 655 Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke  
 656 Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Hui-  
 657 long Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi  
 658 Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li,  
 659 Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin  
 660 Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming  
 661 Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang,  
 662 Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian,  
 663 Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia,  
 664 Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang,  
 665 Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming  
 666 Pan, Ying Zhang, Yingli Wang, Yingru Li, Yipeng Su, Yipeng Geng, Yitong Zhu, Yongkun Yang,  
 667 Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen  
 668 Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zi-  
 669 han Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wen-  
 670 guang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and  
 671 Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint*,  
 672 abs/2508.06471, 2025a.

672 Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen,  
 673 Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong,  
 674 Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao,  
 675 Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang  
 676 Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu,  
 677 Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin,  
 678 Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao  
 679 Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin  
 680 Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu,  
 681 Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe  
 682 Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo  
 683 Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi,  
 684 Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng  
 685 Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang,  
 686 Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang,  
 687 Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu,  
 688 Xinzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing  
 689 Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie  
 690 Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao,  
 691 Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang  
 692 Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang,  
 693 Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng  
 694 Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou,  
 695 Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi K2: Open agentic intelligence.  
 696 *arXiv preprint*, abs/2507.20534, 2025b.

696 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi,  
 697 Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language  
 698 models. In *Advances in Neural Information Processing Systems 35*, New Orleans, LA, 2022.

699 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc  
 700 Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models,  
 701 2023.

702 Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston.  
703 Neural text generation with unlikelihood training. *arXiv preprint*, abs/1908.04319, 2019.  
704

705 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang  
706 Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint*, abs/2505.09388,  
707 2025a.

708 Haotong Yang, Yi Hu, Shijia Kang, Zhouchen Lin, and Muhan Zhang. Number cookbook: Number  
709 understanding of language models and how to improve it. In *Proceedings of the 13th International  
710 Conference on Learning Representations*, Singapore, 2025b.

711 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.  
712 React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th Interna-  
713 tional Conference on Learning Representations*, Kigali, Rwanda, 2023.

714

715 Haizi Yu, Igor Mineyev, Lav R. Varshney, and James A. Evans. Learning from one and only one  
716 shot. *arXiv preprint*, abs/2201.08815, 2022.

717

718 Yiming Zhang, Shi Li, and Yang Liu. On the robustness of chinese text processing pipelines. *Find-  
719 ings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 3548–3560,  
720 2022.

721 Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivas Iyer, Jianfeng Sun, Yuning Mao, Xuezhe Ma, Avia  
722 Efrat, Ping Yu, Lili Yu, et al. Mixture-of-experts meets instruction tuning: A winning combination  
723 for large language models. *arXiv preprint*, abs/2305.14705, 2023.

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756  
 757 Table 4: Performance of Qwen3-30B-A3B-Instruct-2507 and Qwen3-30B-A3B-Thinking-2507 on  
 758 the strict numerical sequence repetition task. The table shows the number of correct reproductions  
 759 and accuracy for each sequence length range.

760 761 Size	762 763 764 765 Total	766 Qwen3-30B-A3B-Instruct Model		767 Qwen3-30B-A3B-Thinking-2507	
		768 # Correct	769 Accuracy	770 # Correct	771 Accuracy
S: 2-32	50	50	100.00%	50	100.00%
M: 33-128	50	50	100.00%	36	72.00%
L: 129-256	50	50	100.00%	16	32.00%
XL: 257-512	50	7	14.00%	0	0.00%
XXL: 513-1024	50	0	0.00%	0	0.00%
<b>Overall</b>	<b>250</b>	<b>157</b>	<b>62.80%</b>	<b>102</b>	<b>40.80%</b>

## 768 769 A THE USE OF LARGE LANGUAGE MODELS

770  
 771 We employed LLMs for bug detection in code. Additionally, LLMs were utilized to refine and polish  
 772 the manuscript content based on specific requirements. All LLM-generated content, including code  
 773 and textual revisions, underwent thorough review and validation by the authors to ensure accuracy,  
 774 quality, and alignment with our research objectives.

## 775 776 B THE EXPERIMENT OF REPETITION DILEMMA

777  
 778 To evaluate the capability of LLMs on repetition, we conducted a preliminary experiment on “strict  
 779 numerical sequence repetition.” The task requires a model to reproduce a given numerical sequence  
 780 exactly, without any additions, omissions, or alterations. We designed a testing framework with  
 781 progressively increasing difficulty by dividing sequence lengths into five ranges: 2–32, 33–128,  
 782 129–256, 257–512, and 513–1024. For each range, 50 unique samples were randomly generated.  
 783 Each sequence consisted of numbers with three decimal places, drawn uniformly from the interval  
 784 [-10, 10]. Models were prompted to return the output in a strict JSON array format (e.g., [1.234, -  
 785 5.678, 9.012]), prohibiting any extraneous characters, spaces, or line breaks. A response was judged  
 786 as correct only if it was an exact string match to the ground truth sequence.

787 The experiment was performed on two variants of the Qwen3 model: Qwen3-30B-A3B-Instruct-  
 788 2507 and Qwen3-30B-A3B-Thinking-2507. To ensure deterministic and stable outputs, the decod-  
 789 ing temperature was set to 0. Any deviation in format or content from the expected output was  
 790 classified as an error.

791 The results, presented in Table 4, reveal a strong “repetition dilemma” in both models. The Qwen3-  
 792 30B-A3B-Instruct-2507 variant performed flawlessly on sequences up to 256 numbers, achieving  
 793 100% accuracy. However, its performance collapsed to just 14% accuracy in the 257–512 range (XL)  
 794 and failed completely on the longest sequences (XXL). Counter-intuitively, the Qwen3-30B-A3B-  
 795 Thinking-2507 variant, despite its designation, demonstrated inferior overall performance (40.80%  
 796 vs. 62.80%). Its accuracy began to degrade significantly on medium-length sequences (M and L),  
 797 falling far short of its instruct-tuned counterpart.

798 These findings highlight significant architectural or attentional limitations in current LLMs for tasks  
 799 demanding precise, long-sequence replication. Such failures may stem from compounding errors in  
 800 the attention mechanism, effective context window constraints, or biases in the training data. The  
 801 inferior performance of the “Thinking” variant is particularly noteworthy. It suggests that for rote,  
 802 mechanical tasks that do not require reasoning, the cognitive overhead or architectural modifications  
 803 intended to facilitate complex thought may act as a source of noise, thereby degrading performance  
 804 on simple memorization and reproduction.

## 805 806 C DATASET DETAILS

807  
 808 This appendix provides detailed descriptions of the datasets used to evaluate LLMs’ long numerical  
 809 sequence processing capabilities.

810 C.1 SYNTHETIC DATASET  
811812 The synthetic dataset  $D_{\text{syn}}$  consists of 1,200 samples across six task types, each containing 200  
813 samples. The sequences are categorized into four length intervals:  
814

- **S (Short):** [2, 32] numbers (50 samples per task)
- **M (Medium):** (32, 128] numbers (50 samples per task)
- **L (Large):** (128, 256] numbers (50 samples per task)
- **XL (Extra-large):** (256, 512] numbers (50 samples per task)

821 C.1.1 TASK TYPES  
822823 **max-int:** Identify the index (0-based) of the maximum integer in an integer sequence. Example:  
824

```

1 {
2   "task_type": "max_int",
3   "answer": "7",
4   "ts": [3, 2, 2, 0, 3, 0, 2, 5, 0, 0, 1, 0, 1, 0, 1, 2, 0, 1, 4]
5 }
```

830 The maximum value is 5 at index 7.  
831832 **min-int:** Identify the index (0-based) of the minimum integer in an integer sequence. Example:  
833

```

1 {
2   "task_type": "min_int",
3   "answer": "19",
4   "ts": [6, 9, 7, 6, 7, 7, 6, 6, 6, 7, 9, 8, 7, 6, 6, 8, 8, 8, 9, 2, 9, 9, 6, 9, 6, 8
5   , 6, 9, 6]
```

839 The minimum value is 2 at index 19.  
840842 **max-float:** Identify the index (0-based) of the maximum floating-point number in a sequence. Simi-  
843 lar to max-int but with floating-point numbers.  
844845 **min-float:** Identify the index (0-based) of the minimum floating-point number in a sequence. Similar  
846 to min-int but with floating-point numbers.  
847848 **indexing:** Determine the position of the last occurrence of 1 in a binary sequence. Example:  
849

```

1 {
2   "task_type": "indexing",
3   "answer": "8",
4   "ts": [1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

854 The last occurrence of 1 is at index 8.  
855856 **counting:** Count the total number of 1s in a binary sequence. Example:  
857

```

1 {
2   "task_type": "counting",
3   "answer": "4",
4   "ts": [1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

863 There are 4 occurrences of 1 in the sequence.  
864

864  
865

## C.2 REAL DATASET

866  
867  
868

The real dataset  $D_{\text{real}}$  consists of 800 samples across four categories, each containing 200 samples. These tasks are based on practical numerical reasoning scenarios.

869

## C.2.1 TASK CATEGORIES

870

**number-string:** Count numerals in alphanumeric sequences. Example:

871  
872  
873  
874  
875  
876  
877  
878

```

1  {
2    "question": "How many numbers are there in the string? Note that a sequence like
3      'a243b' counts as a single number.",
4    "struct_data": "effV2xM8hF5vcNg18xrTCmbD6sEM38tiK4Nn2vem14f698o7Lo",
5    "answer": 11
}
```

879

This task requires parsing mixed alphanumeric strings to identify and count distinct numerical sequences.

880

**number-list:** Perform logical reasoning over numerical sequences with multiple-choice questions. Example:

881  
882  
883

```

1  {
2    "question": "Which index holds the greatest number in the list between the indices
3      20 and 80? Options: A: 40, B: 75, C: 53, D: 58, E: 48, F: 44, G: 60, H: 31",
4    "struct_data": [1372.31, -3479.74, 1046, "..."],
5    "answer": "H"
}
```

884  
885  
886  
887  
888  
889  
890  
891  
892

These tasks involve complex reasoning operations such as finding extrema within specific ranges, identifying patterns, or performing conditional operations.

893

**stock:** Answer questions about financial time series data. Example:

894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917

```

{
  "question": "How many days had a volume over 15,000 between 2024-10-15 and
2024-10-25? Options: A: 3, B: 5, C: 7, D: 9",
  "struct_data": [
    {"date": "2024-10-15", "close_price": 52.56, "volume": 24421, "...": "..."}, 
    {"date": "2024-10-16", "close_price": 52.80, "volume": 19962, "...": "..."}, 
    {"date": "2024-10-17", "close_price": 53.11, "volume": 19210, "...": "..."}, 
    {"date": "2024-10-18", "close_price": 55.11, "volume": 25238, "...": "..."}, 
    ...
  ],
  "answer": "C"
}
```

Listing 1: Stock example (truncated)

This task involves analyzing real-world financial time series data with questions about trading volumes, price movements, and temporal patterns.

**weather:** Answer questions about meteorological time series data. Example:

```

{
  "question": "On which date was the temperature lastly above 5 degrees between
2024-11-10 and 2024-11-20? Options: A: 2024-11-11, B: 2024-11-14, C:
2024-11-12, D: 2024-11-13",
  "struct_data": [
    {"date": "2024-11-10", "temperature_2m": 5.99, "precipitation": 0.0, "...":
    "..."}, 
    {"date": "2024-11-11", "temperature_2m": 5.51, "precipitation": 0.0, "...":
    "..."}, 
    {"date": "2024-11-12", "temperature_2m": 5.10, "precipitation": 0.0, "...":
    "..."}, 
    ...
  ],
  "answer": "D"
}
```

```

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
306
```

972  
973 Table 6: Average performance comparison of different tasks (Vanilla vs SepNS). **Green** and **red**  
974 indicate improvement and degradation in performance metrics, respectively.

975 976 Task	977 Answer Rate ( $\uparrow$ )			978 Accuracy ( $\uparrow$ )			979 Response Len ( $\downarrow$ )			980 Total Len ( $\downarrow$ )		
	981 Vanilla	982 SepNS	983 Incr.	984 Vanilla	985 SepNS	986 Incr.	987 Vanilla	988 SepNS	989 Incr.	990 Vanilla	991 SepNS	992 Incr.
993 counting	994 100.0%	995 100.0%	996 <b>+0.0%</b>	997 42.7%	998 76.7%	999 <b>+79.6%</b>	1000 3004	1001 2106	1002 <b>-29.9%</b>	1003 3594	1004 2740	1005 <b>-23.8%</b>
1006 indexing	1007 100.0%	1008 100.0%	1009 <b>+0.0%</b>	1010 38.8%	1011 86.6%	1012 <b>+123.0%</b>	1013 4035	1014 2304	1015 <b>-42.9%</b>	1016 4573	1017 2909	1018 <b>-36.4%</b>
1019 max-float	1020 100.0%	1021 100.0%	1022 <b>+0.0%</b>	1023 63.9%	1024 81.0%	1025 <b>+26.8%</b>	1026 3459	1027 3603	1028 <b>+4.2%</b>	1029 4544	1030 4843	1031 <b>+6.6%</b>
1032 max-int	1033 100.0%	1034 100.0%	1035 <b>+0.0%</b>	1036 75.3%	1037 92.4%	1038 <b>+22.7%</b>	1039 2242	1040 1744	1041 <b>-22.2%</b>	1042 2741	1043 2334	1044 <b>-14.9%</b>
1045 min-float	1046 100.0%	1047 100.0%	1048 <b>+0.0%</b>	1049 63.1%	1050 79.3%	1051 <b>+25.7%</b>	1052 3676	1053 3722	1054 <b>+1.2%</b>	1055 4764	1056 4945	1057 <b>+3.8%</b>
1058 min-int	1059 100.0%	1060 100.0%	1061 <b>+0.0%</b>	1062 74.3%	1063 93.0%	1064 <b>+25.1%</b>	1065 2338	1066 1829	1067 <b>-21.8%</b>	1068 2843	1069 2425	1070 <b>-14.7%</b>
1071 number-string	1072 96.6%	1073 99.0%	1074 <b>+2.5%</b>	1075 81.6%	1076 81.7%	1077 <b>+0.1%</b>	1078 1855	1079 1569	1080 <b>-15.4%</b>	1081 2059	1082 1803	1083 <b>-12.4%</b>
1084 number-list	1085 78.2%	1086 79.1%	1087 <b>+1.1%</b>	1088 36.3%	1089 36.7%	1090 <b>+0.9%</b>	1091 2881	1092 4205	1093 <b>+46.0%</b>	1094 5643	1095 7526	1096 <b>+33.4%</b>
1097 stock	1098 64.1%	1099 73.6%	1100 <b>+14.7%</b>	1101 13.2%	1102 27.4%	1103 <b>+107.6%</b>	1104 5285	1105 3721	1106 <b>-29.6%</b>	1107 13328	1108 13173	1109 <b>-1.2%</b>
1110 weather	1111 66.0%	1112 76.8%	1113 <b>+16.3%</b>	1114 26.7%	1115 44.6%	1116 <b>+67.1%</b>	1117 4972	1118 3428	1119 <b>-31.1%</b>	1120 11901	1121 11642	1122 <b>-2.2%</b>
1123 Average	1124 90.5%	1125 92.8%	1126 <b>+2.6%</b>	1127 51.6%	1128 69.9%	1129 <b>+35.6%</b>	1130 3375	1131 2823	1132 <b>-16.3%</b>	1133 5599	1134 5434	1135 <b>-2.9%</b>

985  
986 Table 7: Average performance comparison of different models (Vanilla vs SepNS). **Green** and **red**  
987 indicate improvement and degradation in performance metrics, respectively.

990 991 Model	992 Answer Rate ( $\uparrow$ )			993 Accuracy ( $\uparrow$ )			994 Response Len ( $\downarrow$ )			995 Total Len ( $\downarrow$ )		
	996 Vanilla	997 SepNS	998 Incr.	999 Vanilla	1000 SepNS	1001 Incr.	1002 Vanilla	1003 SepNS	1004 Incr.	1005 Vanilla	1006 SepNS	1007 Incr.
1008 QwQ-32B	1009 72.7%	1010 75.6%	1011 <b>+4.0%</b>	1012 34.2%	1013 57.8%	1014 <b>+69.0%</b>	1015 7993	1016 6487	1017 <b>-18.9%</b>	1018 10984	1019 9953	1020 <b>-9.4%</b>
1021 Qwen3-30B-A3B	1022 80.6%	1023 90.2%	1024 <b>+11.9%</b>	1025 54.3%	1026 80.7%	1027 <b>+48.6%</b>	1028 6403	1029 5028	1030 <b>-21.5%</b>	1031 9392	1032 8492	1033 <b>-9.6%</b>
1034 Qwen3-8B	1035 79.4%	1036 87.3%	1037 <b>+9.9%</b>	1038 45.5%	1039 69.6%	1040 <b>+53.0%</b>	1041 6878	1042 6001	1043 <b>-12.7%</b>	1044 9866	1045 9466	1046 <b>-4.1%</b>
1047 DeepSeek-V3	1048 100.0%	1049 100.0%	1050 <b>+0.0%</b>	1051 45.7%	1052 50.9%	1053 <b>+11.4%</b>	1054 720	1055 882	1056 <b>+22.5%</b>	1057 2762	1058 3404	1059 <b>+23.2%</b>
1060 DeepSeek-R1	1061 99.9%	1062 99.9%	1063 <b>+0.0%</b>	1064 61.1%	1065 70.5%	1066 <b>+15.4%</b>	1067 5438	1068 3798	1069 <b>-30.1%</b>	1070 7481	1071 6276	1072 <b>-16.1%</b>
1073 Claude-3.7-Sonnet	1074 99.8%	1075 99.9%	1076 <b>+0.1%</b>	1077 57.3%	1078 79.1%	1079 <b>+38.0%</b>	1080 444	1081 535	1082 <b>+20.6%</b>	1083 2553	1084 3182	1085 <b>+24.7%</b>
1086 Gemini-2.5-Pro	1087 83.4%	1088 84.6%	1089 <b>+1.4%</b>	1090 58.9%	1091 79.1%	1092 <b>+34.3%</b>	1093 654	1094 822	1095 <b>+25.6%</b>	1096 3671	1097 4424	1098 <b>+20.5%</b>
1099 GPT-4.1	1100 99.6%	1101 99.5%	1102 <b>-0.1%</b>	1103 61.0%	1104 82.7%	1105 <b>+35.6%</b>	1106 1226	1107 1115	1108 <b>-9.1%</b>	1109 2452	1110 2229	1111 <b>-9.1%</b>
1112 GPT-4o	1113 99.0%	1114 98.6%	1115 <b>-0.4%</b>	1116 46.4%	1117 59.1%	1118 <b>+27.4%</b>	1119 616	1120 740	1121 <b>+20.0%</b>	1122 1231	1123 1478	1124 <b>+20.1%</b>
1125 Average	1126 90.5%	1127 92.8%	1128 <b>+2.6%</b>	1129 51.6%	1130 69.9%	1131 <b>+35.6%</b>	1132 3375	1133 2823	1134 <b>-16.3%</b>	1135 5599	1136 5434	1137 <b>-2.9%</b>

1000  
1001 Rate gains (+1.1%), and number-string tasks that maintain similar performance levels (+0.1% accuracy, +2.5% Answer Rate). Concurrently, the Response Length for these tasks decreased significantly (e.g., indexing: -42.9%, counting: -29.9%), demonstrating more concise and accurate model output. The max/min-int type tasks also achieved improvements exceeding 22% from already relatively high baseline Accuracy (75%). A critical outlier is the number-list task, where Accuracy remained almost unchanged under SepNS (+0.9%), yet Response Length increased sharply (+46.0%). This suggests that the current design of the SepNS scheme may be ineffective at addressing the core challenges of this task, instead introducing unnecessary verbose output.

1002  
1003 Illustrated in Table 7, this experiment compared the performance of the baseline method (Vanilla) and the SepNS scheme across a range of mainstream large language models. Evaluation metrics included Answer Rate, Accuracy, Response Length, and Total Length. The results demonstrate that the SepNS scheme significantly outperforms the baseline in the vast majority of cases. Overall, SepNS improved the average Answer Rate (+2.6%) and average Accuracy (+35.6%), with the gain in Accuracy being particularly remarkable. Furthermore, the scheme effectively reduced the average Response Length (-16.3%) and Total Length (-2.9%), indicating that it enhances not only performance but also output efficiency.

1004  
1005 Specifically, in terms of Accuracy, all models showed improvement, with the most significant gains observed in QwQ-32B (+69.0%), Qwen3-8B (+53.0%), and Qwen3-30B-A3B (+48.6%). Regarding output efficiency, the generated length was substantially reduced for most models, such as QwQ-32B (Response Length: -18.9%) and DeepSeek-R1 (Response Length: -30.1%). However, some anomalies were observed: the Response Length increased for DeepSeek-V3 (+22.5%), Claude-3.7-Sonnet (+20.6%), Gemini-2.5-Pro (+25.6%), and GPT-4o (+20.0%) under SepNS. This may stem from specific interactions between their inherent reasoning patterns and the structured output requirements, though all of these models still achieved positive gains in Accuracy (ranging from +11.4% to +38.0%).

1026  
1027 Table 8: Accuracy of Qwen3-30B-A3B applied vanilla and SepNS across separate interval compar-  
1028 ing Instruct (Qwen3-30B-A3B-Instruct-2507) and Reasoning (Qwen3-30B-A3B-Thinking-2507)  
1029 model. **Bold** and underlined values denote the highest and second-highest scores, respectively.

1030 1031 Task	1032 Accuracy of Instruct Model ( $\uparrow$ )				1033 Accuracy of Reasoning Model ( $\uparrow$ )					
	1034 vanilla	1035 4	1036 8	1037 16	1038 32	1039 vanilla	1040 4	1041 8	1042 16	1043 32
max-int	82.5%	81.5%	85.0%	<b>94.5%</b>	<u>91.5%</u>	88.0%	<b>100.0%</b>	<b>100.0%</b>	<b>100.0%</b>	<b>100.0%</b>
max-float	71.5%	74.0%	75.0%	<b>82.5%</b>	<u>82.0%</u>	73.0%	<u>96.0%</u>	<b>97.5%</b>	<b>97.5%</b>	<b>97.5%</b>
min-int	76.5%	78.5%	84.0%	<u>86.0%</u>	<b>89.5%</b>	82.0%	99.0%	<b>100.0%</b>	<u>99.5%</u>	<b>100.0%</b>
min-float	74.0%	80.0%	77.5%	<u>82.0%</u>	<b>83.0%</b>	65.5%	89.0%	<b>92.0%</b>	<b>95.0%</b>	90.5%
indexing	29.0%	80.0%	83.5%	<b>90.5%</b>	<u>85.5%</u>	34.5%	<b>100.0%</b>	<b>99.5%</b>	99.0%	98.0%
counting	32.5%	65.0%	<u>74.0%</u>	<b>76.0%</b>	73.0%	40.0%	89.0%	<b>97.5%</b>	<b>99.0%</b>	95.5%
number-string	<b>98.0%</b>	80.5%	<b>98.5%</b>	97.5%	95.0%	99.0%	67.0%	<b>99.5%</b>	<u>99.5%</u>	<b>100.0%</b>
number-list	38.5%	37.5%	<u>41.0%</u>	<b>43.0%</b>	<u>43.0%</u>	39.0%	<u>45.0%</u>	<b>46.5%</b>	42.0%	41.0%
stock	19.0%	36.5%	<u>40.5%</u>	<b>41.0%</b>	35.5%	8.5%	<u>50.5%</u>	<b>57.5%</b>	45.0%	41.5%
weather	24.0%	<u>24.5%</u>	19.5%	17.5%	<b>30.0%</b>	29.0%	<b>75.0%</b>	<u>71.5%</u>	69.0%	67.5%
Average	54.6%	63.8%	67.8%	<b>71.0%</b>	<u>70.8%</u>	55.9%	81.0%	<b>86.2%</b>	<u>84.5%</u>	83.1%

1040  
1041 Table 9: Answer rate and accuracy of Qwen3-30B-A3B-Instruct applied SepNS across different  
1042 separator symbols. **Bold** and underlined values denote the highest and second-highest scores, re-  
1043 spectively.

1044 1045 Task	1046 Answer Rate ( $\uparrow$ )				1047 Accuracy ( $\uparrow$ )			
	1048 CR(\r)	1049 CRLF(\r\n)	1050 Backslash(\\\)	1051 LF(\n)	1052 CR(\r)	1053 CRLF(\r\n)	1054 Backslash(\\\)	1055 LF(\n)
max-int	100.0%	100.0%	100.0%	100.0%	93.0%	89.0%	92.0%	85.0%
max-float	100.0%	100.0%	100.0%	100.0%	79.0%	74.5%	76.0%	75.0%
min-int	100.0%	100.0%	100.0%	100.0%	87.0%	87.0%	91.5%	84.0%
min-float	100.0%	100.0%	100.0%	100.0%	77.5%	75.5%	72.0%	77.5%
indexing	100.0%	100.0%	100.0%	100.0%	87.0%	90.0%	92.5%	83.5%
counting	100.0%	100.0%	100.0%	100.0%	73.0%	75.0%	73.0%	74.0%
Average	<b>100.0%</b>	<b>100.0%</b>	<b>100.0%</b>	<b>100.0%</b>	<u>82.8%</u>	81.8%	<b>82.8%</b>	79.8%
number_string	100.0%	100.0%	100.0%	100.0%	99.0%	98.0%	98.5%	98.5%
number_list	59.5%	69.0%	60.0%	67.0%	39.5%	41.5%	35.5%	41.0%
stock	76.5%	82.0%	76.0%	80.5%	38.0%	33.5%	38.0%	40.5%
weather	97.5%	98.0%	99.5%	98.5%	19.5%	17.0%	9.0%	19.5%
Average	83.4%	<b>87.2%</b>	83.9%	<u>86.5%</u>	<u>49.0%</u>	47.5%	45.2%	<b>49.9%</b>

## 1058 E.2 TABLE RESULTS FOR RQ2

1059 We provide detailed experimental results for Figure 4 in Table 8 and Table 9.

## 1062 E.3 RAW ACCURACY RESULTS OF SYNTHETIC DATASET

1064 Table 10 reports the baseline performance, without applying SepNS, across the six synthetic  
1065 sequence-based tasks introduced in Section 4.1.1: *min-int*, *max-int*, *counting*, *min-float*,  
1066 and *max-float*. For each task, datasets are categorized by input sequence length into four sizes: small  
1067 (S), medium (M), large (L), and extra-large (XL).

1068 Across all models and tasks, we observe a consistent and pronounced performance degradation  
1069 as sequence length increases. While most models achieve near-perfect accuracy on the S and M  
1070 settings, accuracy drops sharply for larger inputs, with many models falling below 10% on the XL  
1071 datasets. This trend persists even for models known for strong reasoning capabilities, underscoring  
1072 a general limitation of current LLMs in processing long numerical sequences.

1073 Table 11 summarizes the accuracy gains achieved by enhancing LLMs with SepNS, compared to  
1074 the baseline results in Table 10. The improvements are particularly pronounced for tasks involving  
1075 large (L) and extra-long (XL) input sequences, where several models exhibit gains exceeding 0.8  
1076 in accuracy. While performance in the small (S) and medium (M) ranges is generally stable—with  
1077 occasional minor decreases—the consistent boost for L and XL sequences highlights SepNS’s ef-  
1078 fectiveness in mitigating context-length degradation. These trends are evident across diverse tasks,  
1079 including min/max integer and float extraction, counting, and indexing, underscoring the robustness  
of SepNS across both discrete and continuous input domains.

1080  
1081 Table 10: Accuracy of vanilla method on sequence-based tasks. The results highlight significant  
1082 performance variation among the models and reveal a consistent trend of decreasing accuracy as the  
1083 input sequence becomes longer.

Model	min-int				max-int				counting			
	S	M	L	XL	S	M	L	XL	S	M	L	XL
Qwen3-8B	100%	92%	60%	20%	100%	96%	80%	32%	96%	28%	0%	0%
Qwen3-30B-A3B	100%	100%	88%	48%	100%	98%	92%	52%	100%	42%	4%	0%
QwQ-32B	100%	62%	18%	6%	100%	62%	38%	2%	94%	20%	0%	0%
DeepSeek-V3	100%	98%	74%	32%	96%	98%	64%	28%	98%	66%	28%	4%
DeepSeek-R1	100%	100%	98%	76%	100%	100%	94%	74%	100%	86%	34%	10%
Claude-3.7-Sonnet	100%	86%	70%	26%	100%	90%	64%	26%	94%	70%	32%	10%
Gemini-2.5-Pro	98%	94%	92%	72%	98%	98%	92%	76%	96%	80%	46%	10%
GPT-4.1	100%	94%	70%	46%	100%	94%	72%	48%	96%	34%	8%	2%
GPT-4o	100%	90%	44%	20%	100%	88%	42%	14%	92%	40%	16%	2%
Model	min-float				max-float				indexing			
	S	M	L	XL	S	M	L	XL	S	M	L	XL
Qwen3-8B	100%	94%	36%	2%	100%	88%	30%	2%	96%	16%	0%	0%
Qwen3-30B-A3B	100%	98%	64%	10%	100%	100%	78%	24%	100%	34%	4%	0%
QwQ-32B	100%	26%	0%	0%	100%	32%	8%	0%	98%	10%	0%	0%
DeepSeek-V3	100%	86%	48%	20%	100%	98%	42%	18%	98%	78%	18%	0%
DeepSeek-R1	100%	100%	94%	92%	100%	100%	100%	96%	100%	72%	26%	4%
Claude-3.7-Sonnet	100%	84%	42%	26%	100%	92%	32%	20%	100%	46%	16%	2%
Gemini-2.5-Pro	98%	100%	28%	10%	98%	90%	42%	4%	100%	60%	28%	6%
GPT-4.1	100%	92%	56%	40%	100%	94%	58%	44%	100%	44%	12%	2%
gpt-4o-2024-08-06	100%	78%	32%	16%	100%	72%	24%	12%	96%	30%	2%	0%

1101  
1102 Table 11: Accuracy improvement of SepNS over vanilla method. Results are broken down by  
1103 LLMs and input sequence length for various sequence-based tasks. Higher values indicate a greater  
1104 improvement.

Model	min-int				max-int				counting			
	S	M	L	XL	S	M	L	XL	S	M	L	XL
Qwen3-8B	0%	8%	40%	78%	0%	4%	20%	66%	0%	66%	76%	54%
Qwen3-30B-A3B	0%	-2%	12%	52%	0%	2%	8%	46%	0%	50%	84%	82%
QwQ-32B	0%	38%	80%	70%	0%	38%	62%	92%	6%	76%	76%	22%
Claude-3.7-Sonnet	0%	12%	28%	68%	0%	10%	34%	68%	6%	30%	56%	12%
DeepSeek-V3	-12%	-6%	6%	10%	-6%	-22%	8%	10%	-10%	20%	-8%	0%
DeepSeek-R1	-8%	0%	2%	24%	-6%	0%	4%	26%	-6%	14%	66%	84%
Gemini-2.5-Pro	2%	4%	8%	28%	0%	2%	8%	24%	4%	20%	52%	74%
GPT-4.1	0%	0%	24%	36%	0%	4%	20%	42%	2%	44%	54%	42%
GPT-4o	0%	-8%	34%	50%	0%	4%	20%	44%	8%	52%	16%	6%
Model	min-float				max-float				indexing			
	S	M	L	XL	S	M	L	XL	S	M	L	XL
QwQ-32B	-2%	46%	2%	0%	0%	50%	8%	0%	0%	90%	84%	52%
Qwen3-30B-A3B	0%	0%	22%	58%	0%	-2%	16%	46%	-4%	64%	92%	98%
Qwen3-8B	-4%	-2%	26%	26%	0%	10%	40%	40%	2%	78%	96%	86%
Claude-3.7-Sonnet	0%	12%	46%	58%	0%	2%	56%	70%	0%	46%	70%	90%
DeepSeek-V3	-16%	0%	28%	10%	-22%	-14%	26%	12%	-8%	-2%	38%	36%
DeepSeek-R1	-8%	0%	6%	6%	-10%	0%	0%	4%	-10%	28%	74%	96%
Gemini-2.5-Pro	0%	-2%	68%	80%	0%	8%	50%	78%	-2%	34%	58%	92%
GPT-4.1	-2%	0%	30%	48%	0%	0%	38%	40%	0%	50%	88%	96%
GPT-4o	-6%	10%	20%	30%	-6%	12%	38%	22%	-2%	26%	52%	36%

#### E.4 RAW RESPONSE LENGTH RESULTS OF SYNTHETIC DATASET

As shown in Table 12, SepNS consistently reduces response length for well-aligned models, with its benefits becoming particularly pronounced as input sequences grow longer.

The true strength of SepNS is unlocked when processing long contexts, a critical challenge for LLMs. This is powerfully illustrated by models like GPT-4.1, which achieved dramatic response length reductions of up to 78.9% and 89.6% on extra-large (XL) sequences for ‘min-int’ and ‘indexing’ tasks, respectively. Similarly, the Qwen series models consistently benefit from SepNS on medium (M) to XL sequences, especially on complex tasks like ‘counting’ and ‘indexing’, where token savings often exceed 50%. Furthermore, the QwQ-32B model shows exceptional affinity for

1134

1135 Table 12: Percentage change in response length (RL) using SepNS over baselines. Results are  
1136 broken down by LLM and input sequence length for various sequence-based tasks in a zero-shot  
1137 setting. More negative values signify a greater improvement (i.e., token number reduction).

1138	Model	min-int				max-int				counting			
		S	M	L	XL	S	M	L	XL	S	M	L	XL
1140	Qwen3-8B	57.3%	-3.4%	-25.4%	-37.0%	78.1%	0.6%	-21.7%	-39.3%	67.6%	-40.4%	-36.8%	-18.1%
1141	QwQ-32B	-14.0%	-52.0%	-38.0%	-27.0%	-10.1%	-51.5%	-40.4%	-37.3%	-59.6%	-67.3%	-35.6%	-5.6%
1142	Qwen3-30B-A3B	93.9%	20.4%	-22.7%	-45.9%	109.7%	5.2%	-23.2%	-43.4%	42.8%	-57.3%	-57.2%	-37.9%
1143	Claude-3.7-Sonnet	-4.3%	42.2%	0.6%	39.9%	-10.9%	30.8%	-11.6%	12.9%	-27.6%	5.0%	41.2%	66.9%
1144	DeepSeek-V3	-31.4%	-19.9%	-1.8%	5.2%	-22.8%	-13.5%	28.5%	35.6%	12.6%	6.4%	39.9%	178.4%
1145	DeepSeek-R1	80.6%	77.3%	261.6%	565.7%	96.6%	33.5%	235.7%	252.4%	-6.8%	55.7%	126.3%	62.9%
1146	Gemini-2.5-Pro	-3.8%	-2.8%	-2.5%	-15.5%	-9.5%	-4.9%	-6.9%	2.3%	-1.3%	-3.7%	14.4%	140.1%
1147	gGPT-4.1	-0.4%	25.5%	-23.7%	-78.9%	0.8%	38.8%	-23.8%	-76.8%	3.5%	-31.0%	-40.2%	-60.6%
1148	GPT-4o	13.5%	29.2%	15.9%	-24.2%	4.9%	37.4%	2.3%	-22.0%	-4.2%	-3.9%	-17.9%	76.9%
1149	Model	min-float				max-float				indexing			
		S	M	L	XL	S	M	L	XL	S	M	L	XL
1150	Qwen3-8B	36.2%	3.6%	-7.7%	-4.5%	48.6%	4.1%	-9.1%	-7.3%	-1.5%	-54.9%	-45.3%	-31.3%
1151	Qwen3-30B-A3B	63.6%	4.0%	-8.9%	-17.4%	82.4%	25.0%	-1.6%	-17.9%	34.6%	-52.2%	-48.6%	-47.5%
1152	QwQ-32B	-13.7%	-14.6%	-0.3%	0.0%	-8.8%	-24.3%	-1.0%	0.0%	-39.0%	-59.6%	-31.4%	-13.8%
1153	Claude-3.7-Sonnet	-11.8%	86.3%	143.5%	126.1%	-20.6%	29.7%	123.5%	162.6%	23.2%	9.4%	5.7%	-45.5%
1154	DeepSeek-V3	15.7%	-5.6%	62.1%	146.2%	47.1%	7.8%	130.1%	154.5%	-10.4%	-18.4%	-3.5%	-19.3%
1155	DeepSeek-R1	54.8%	9.1%	-2.9%	-8.1%	69.3%	5.0%	3.9%	0.6%	133.4%	-53.1%	-65.1%	-72.5%
1156	Gemini-2.5-Pro	-7.9%	6.9%	82.4%	134.8%	-16.9%	10.6%	93.0%	187.4%	21.9%	20.9%	-23.9%	-44.2%
1157	GPT-4.1	13.7%	10.4%	-14.0%	-32.7%	14.6%	4.2%	-13.4%	-17.8%	21.2%	1.9%	-68.1%	-89.6%
1158	GPT-4o	10.9%	31.9%	26.8%	19.1%	3.9%	35.9%	29.1%	29.1%	18.5%	68.3%	49.6%	66.1%

1157 the SepNS format, realizing substantial and consistent token reductions across nearly all evaluated  
1158 conditions, highlighting the method’s potential when paired with a compatible model architecture.1159 While some models, such as Claude-3.7-Sonnet and DeepSeek-R1, occasionally produced more ver-  
1160 bose outputs, this variance appears to reveal more about model-specific training than a limitation of  
1161 SepNS. These models, likely heavily fine-tuned for conversational and descriptive tasks, may inter-  
1162 pret the structured SepNS format as a prompt for explanation rather than direct computation, leading  
1163 to increased verbosity. In contrast, models that respond well to SepNS demonstrate a stronger innate  
1164 capability for structured data processing.

1165

1166 

## F SEPNS ATTENTION MECHANISM: RIGOROUS MATHEMATICAL PROOF

1167 

### F.1 DEFINITIONS AND SETUP

1170 **Definition 1** (Vanilla Attention Sequence). Let  $s = \{a_1, a_2, \dots, a_n\}$  be the original sequence  
1171 without separators, where each  $a_i$  represents a token at position  $i$ .1172 **Definition 2** (SepNS Sequence). Let  $s' = \{a_1, \dots, a_k, sep_1, a_{k+1}, \dots, a_{2k}, sep_2, \dots\}$  be the se-  
1173 quence with separator tokens, where  $sep_j$  denotes the  $j$ -th separator token. The total length of  $s'$  is  
1174  $n'$ .1175 **Definition 3** (Segment Partitioning). For a given separator token  $sep$  at position  $p$ , define  $S_{sep} =$   
1176  $\{i : i < p\}$  as the set of all token positions before the separator  $sep$ .1177 **Definition 4** (Query-Key Similarity). For any positions  $i$  and  $j$ , define the scaled dot-product simi-  
1178 larity as:

1179 
$$\alpha_{i,j} = \frac{\mathbf{Q}_i \cdot \mathbf{K}_j^T}{\sqrt{d_k}}$$

1183 

### F.2 MAIN THEOREM

1185 **Theorem 1** (Cross-Segment Attention Suppression). Let  $i_1 \in S_{sep}$  and  $j \notin S_{sep}$  for some separator  
1186  $sep$ . Then:

1187 
$$\frac{A_{\text{SepNS}}[i_1, j]}{A_{\text{Vanilla}}[i_1, j]} = \frac{Z_{\text{vanilla}}(i_1)}{Z_{\text{SepNS}}(i_1)} \ll 1$$

1188 where  $Z_{\text{vanilla}}(i_1)$  and  $Z_{\text{SepNS}}(i_1)$  are the normalization constants for vanilla and SepNS attention  
 1189 respectively.  
 1190

1191 **F.3 PROOF**  
 1192

1193 **Step 1: Establish Attention Weight Formulations** For vanilla attention:

$$1194 \quad 1195 \quad 1196 \quad A_{\text{vanilla}}[i, j] = \frac{\exp(\alpha_{i,j})}{\sum_{l=1}^n \exp(\alpha_{i,l})} = \frac{\exp(\alpha_{i,j})}{Z_{\text{vanilla}}(i)}$$

1197 For SepNS attention:

$$1198 \quad 1199 \quad 1200 \quad A_{\text{SepNS}}[i, j] = \frac{\exp(\alpha_{i,j})}{\sum_{l=1}^{n'} \exp(\alpha_{i,l})} = \frac{\exp(\alpha_{i,j})}{Z_{\text{SepNS}}(i)}$$

1201 **Step 2: Analyze Normalization Constant Difference** The key insight is that:

$$1203 \quad Z_{\text{SepNS}}(i) = Z_{\text{vanilla}}(i) + \Delta Z(i)$$

1204 where  $\Delta Z(i)$  represents the additional normalization mass contributed by separator tokens:  
 1205

$$1206 \quad 1207 \quad \Delta Z(i) = \sum_{sep \in \text{Separators}} \exp(\alpha_{i,sep})$$

1209 **Step 3: Establish Core Lemma** **Lemma 1** (Separator Attention Asymmetry). For separator token  
 1210  $sep$  and positions  $i_1 \in S_{sep}$ ,  $i_2 \notin S_{sep}$ :

$$1211 \quad \alpha_{i_1,sep} \gg \alpha_{i_2,sep}$$

1213 **Proof of Lemma 1:** By the design principle of separator tokens as summarizers of preceding content,  
 1214 the key vector  $\mathbf{K}_{sep}$  is constructed to have high similarity with query vectors  $\mathbf{Q}_{i_1}$  for  $i_1 \in S_{sep}$   
 1215 and low similarity with  $\mathbf{Q}_{i_2}$  for  $i_2 \notin S_{sep}$ . This follows from the separator's role in capturing the  
 1216 semantic representation of tokens up to its position.

1217 **Step 4: Quantify Cross-Segment Attention Suppression** For  $i_1 \in S_{sep}$  and  $j \notin S_{sep}$ :

$$1220 \quad 1221 \quad 1222 \quad \frac{A_{\text{SepNS}}[i_1, j]}{A_{\text{vanilla}}[i_1, j]} = \frac{Z_{\text{vanilla}}(i_1)}{Z_{\text{SepNS}}(i_1)} = \frac{Z_{\text{vanilla}}(i_1)}{Z_{\text{vanilla}}(i_1) + \Delta Z(i_1)}$$

1223 Since  $i_1 \in S_{sep}$ , by Lemma 1, we have  $\exp(\alpha_{i_1,sep})$  is large, making  $\Delta Z(i_1)$  significant. Therefore:

$$1224 \quad 1225 \quad 1226 \quad \frac{Z_{\text{vanilla}}(i_1)}{Z_{\text{vanilla}}(i_1) + \Delta Z(i_1)} = \frac{1}{1 + \frac{\Delta Z(i_1)}{Z_{\text{vanilla}}(i_1)}} \ll 1$$

1227 **Step 5: Analyze Within-Segment Attention Preservation** For  $i_2 \notin S_{sep}$  and  $j$  in the same  
 1228 segment as  $i_2$ :

1229 By Lemma 1,  $\exp(\alpha_{i_2,sep})$  is small, so  $\Delta Z(i_2) \approx 0$ . Therefore:

$$1232 \quad 1233 \quad \frac{A_{\text{SepNS}}[i_2, j]}{A_{\text{vanilla}}[i_2, j]} = \frac{Z_{\text{vanilla}}(i_2)}{Z_{\text{vanilla}}(i_2) + \Delta Z(i_2)} \approx \frac{Z_{\text{vanilla}}(i_2)}{Z_{\text{vanilla}}(i_2)} = 1$$

1234 **F.4 COROLLARY**  
 1235

1236 **Corollary 1** (Attention Localization). The ratio of cross-segment to within-segment attention de-  
 1237 creases exponentially with the separator's query-key similarity:

$$1239 \quad 1240 \quad \frac{A_{\text{SepNS}}[i_1, j_{\text{cross}}]}{A_{\text{SepNS}}[i_1, j_{\text{within}}]} \propto \frac{A_{\text{vanilla}}[i_1, j_{\text{cross}}]}{A_{\text{vanilla}}[i_1, j_{\text{within}}]} \cdot \exp(-\alpha_{i_1,sep})$$

1241 where  $j_{\text{cross}} \notin S_{sep}$  and  $j_{\text{within}} \in S_{sep}$ .

1242 F.5 CONCLUSION  
12431244 This proof rigorously establishes that separator tokens act as “attention sinks” that systematically  
1245 redirect attention mass from cross-segment positions to within-segment positions. The mechanism  
1246 operates through:1247 **Asymmetric Query-Key Similarity.** Separators exhibit high similarity with tokens in their sum-  
1248 marized segment but low similarity with tokens outside.1249 **Normalization Mass Redistribution.** High separator attention scores increase the denominator for  
1250 tokens in the summarized segment, suppressing their cross-segment attention weights.1252 **Selective Suppression.** Only tokens in the summarized segment experience attention suppression,  
1253 while tokens outside maintain their original attention patterns.1254 This mathematical framework explains how SepNS achieves structured attention boundaries without  
1255 explicit masking, creating localized attention patterns that respect segment boundaries.

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295