SepSeq: A Training-Free Framework for Long Numerical Sequence Processing in LLMs

Jie Sun[1,2], **Yu Liu**[3*], **Lu Han**[3], **Qiwen Deng**[4], **Xiang Shu**[5], **Yang Xiao**[6], **Xingyu Lu**[3], **Jun Zhou**[7], **Pengfei Liu**[2], **Lintao Ma**[8†], **Jiancan Wu**[1,9†], **Xiang Wang**[1†]

[1] University of Science and Technology of China [2] Shanghai Innovation Institute
[3] Nanjing University [4] University of Edinburgh [5] East China Normal University
[6] Hong Kong Polytechnic University [7] Zhejiang University [8] Ocean University of China
[9] Institute of Dataspace, Hefei Comprehensive National Science Center

Abstract

While transformer-based Large Language Models (LLMs) theoretically support massive context windows, they suffer from severe performance degradation when processing long numerical sequences. We attribute this failure to the attention dispersion in the Softmax mechanism, which prevents the model from concentrating attention. To overcome this, we propose **Sep**arate **Seq**uence (**SepSeq**), a training-free, plug-and-play framework to mitigate dispersion by strategically inserting separator tokens. Mechanistically, we demonstrate that separator tokens act as an attention sink, recalibrating attention to focus on local segments while preserving global context. Extensive evaluations on 9 widely-adopted LLMs confirm the effectiveness of our approach: SepSeq yields an average relative accuracy improvement of 35.6% across diverse domains while reducing total inference token consumption by 16.4% on average.

1 Introduction

Transformer-based Large Language Models (LLMs) (Vaswani et al., 2017) now support increasingly long context windows (Anthropic, 2025; Gemini-Team, 2025; Meta-AI, 2025; GLM-4.5-Team et al., 2025; Kimi-Team et al., 2025), yet they remain unreliable on long numerical sequence processing tasks. In these tasks, the input is a sequence of numbers, and the model must answer a natural-language question that requires precise access to the sequence, such as counting, filtering, trend identification, or forecasting-style analysis. Unlike open-ended long-context understanding, these tasks are precision-critical: even a small error in preserving or attending to the sequence can change the final answer substantially.

This limitation is not resolved simply by extending the nominal context window (Hosseini et al., 2025; Kuratov et al., 2024). Prior work has shown that effective context utilization often lags behind the advertised context length, especially when precise retrieval from long inputs is required. The challenge is particularly acute for numerical sequences, where many tokens are locally similar, semantically lightweight in isolation, and individually important to the final answer. While external tools (Qu et al., 2025; Feng et al., 2025; Qian et al., 2025; Zhang et al., 2025) or Program-of-Thought (PoT) (Chen et al., 2023a) prompting can assist downstream computation, they still depend on the model's ability to faithfully preserve and reference the original sequence, leaving the upstream input-processing bottleneck largely unchanged.

In this work, we identify attention dispersion as a practical bottleneck for long numerical sequence processing in LLMs. As the input grows longer, attention mass is distributed across a large number of numerically similar tokens, making it harder for the model to maintain focused access to locally relevant values (Nakanishi, 2025; Velickovic et al., 2024). To mitigate this issue, we propose **Sep**arate **Seq**uence (**SepSeq**), a training-free and plug-and-play input-formatting framework that inserts separator tokens at regular intervals to partition a long numerical sequence into shorter segments.

Although separator tokens have been employed in inference acceleration (Chen et al., 2024) and vision transformers (Darcet et al., 2024), their utility for training-free long-content question answering remains unexplored. Functionally, SepSeq operates by introducing systematic separator insertion to define distinct boundaries, effectively transforming intractable long inputs into segments aligned with the model's effective capacity. This segmentation mitigates the precision degradation highlighted in Figure 1(A). Mechanistically, we show that the inserted separators act as attention sinks that attract subsequent attention and reduce cross-segment interference. This creates a more localized attention

*Project Lead
†Corresponding authors

Findings of the Association for Computational Linguistics: ACL 2026, pages 5522–5537
July 2-7, 2026

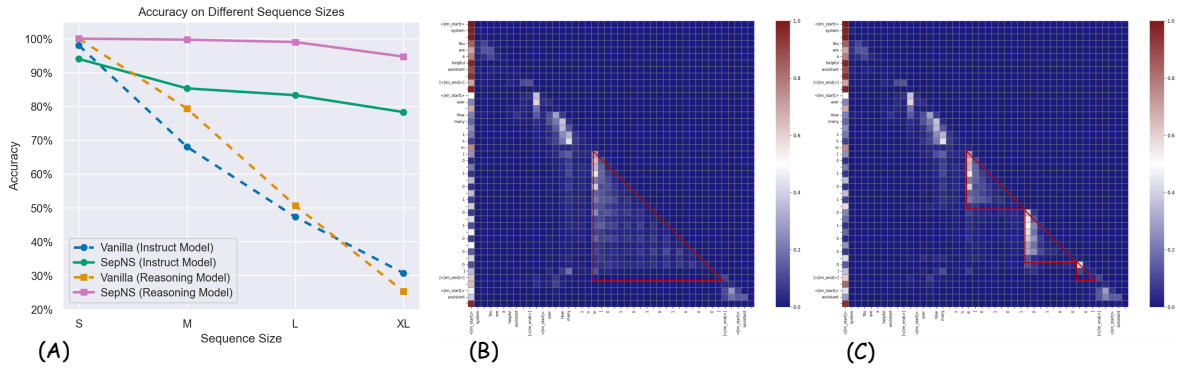

Figure 1: (A) Average accuracy across six synthetic tasks, performance drops sharply with increasing numerical sequence length (S: 2–32, M: 33–128, L: 129–256, XL: 257–512), while SepSeq remains largely unaffected. (B) visualize the attention scores given the input: "... How many 1's in [0 1 0 1 0 1 0 0 0] ...". (C) show the attention scores for the input with segmentation: "... How many 1's in [0 1 0 1\n0 1 0 0\n0] ...". (B) exhibits dispersed attention across the entire sequence while (C) demonstrates segment-focused attention.

pattern, allowing the model to focus on shorter numerical spans while still preserving access to the full sequence, triggering specific heads to concentrate on local segments shown in Figure 1(B,C). Rather than introducing a new attention operator, SepSeq operationalizes a simple inference-time redistribution of attention that is particularly effective for precision-critical numerical inputs.

We conduct comprehensive experiments across 9 high-performance LLMs, evaluating performance on 6 synthetic tasks and 4 real numerical sequence processing domains. Our results demonstrate that SepSeq substantially outperforms baselines, 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 reduces token consumption by 16.4% during inference.

In summary, our main contributions are as follows: **First**, we identify attention dispersion as the bottleneck limiting LLMs' performance on long numerical sequences and visualize how standard Softmax mechanisms fail to produce a sufficiently sharp attention distribution. **Second**, we introduce SepSeq, a plug-and-play, training-free framework that overcomes this limitation by strategically inserting separators to recalibrate attention weights. **Third**, we provide comprehensive empirical validation across diverse datasets, demonstrating that simple input formatting can unlock substantial performance gains (Avg. +35.6%) and reduce 16.4% token consumption during inference.

2 Related Work

Long Context Modeling and Efficiency. While methods like RoPE Scaling (Liu et al., 2023) and YaRN (Peng et al., 2023) have expanded context windows, mere architectural extension does not ensure robust utilization. Recent efficiency-focused works (*e.g.,* StreamingLLM (Xiao et al., 2024), SepLLM (Chen et al., 2024)) address memory bottlenecks but do not fundamentally solve the reasoning degradation. This is evidenced by the "Lost in the Middle" phenomenon (Liu et al., 2024) and the RULER benchmark (Hsieh et al., 2024), which reveal that effective context length often lags far behind nominal window size, particularly for precision-critical tasks.

Representation Bottlenecks in Numerical Processing. Numerical processing presents unique challenges due to the discrete nature of tokenization contrasting with the continuous nature of numerical values. Crucially, recent theoretical analysis (Barbero et al., 2024) suggests that long numerical sequences are prone to "over-squashing", where information is excessively compressed into fixed-size vectors. This compression leads to indistinguishable encodings, preventing the model from resolving specific values within dense sequences. Consequently, models frequently exhibit reasoning failures (Li et al., 2025; Hendrycks et al., 2021). While vocabulary adaptation via retraining presents a viable solution, the extensive requirements for high-quality data, computational resources, and specialized training techniques prove prohibitive for modern large-scale LLMs (Kudo and Richardson, 2018). This bottleneck necessitates the devel-

opment of lightweight, inference-time solutions.

The Transcription Barrier in Tool Use. A prevailing strategy to mitigate calculation errors involves leveraging external tools (Schick et al., 2023; Chen et al., 2023b; Gao et al., 2023) or autonomous agents (Yao et al., 2023; Shinn et al., 2023). However, this paradigm merely shifts the burden from computational reasoning to textual transcription. Existing studies (Welleck et al., 2019; Zhang et al., 2022) indicate that LLMs struggle to faithfully transcribe long sequences without corruption. In the context of long numerical processing, this limitation manifests as an inability to correctly formulate function calls with the complete, unaltered sequence. Consequently, the efficacy of tool-based or agentic methods remains fundamentally constrained by the model's intrinsic capability to attend to and preserve the raw input sequence.

3 Method

In this section, we begin by establishing the formal problem setting for long numerical sequence processing. Next, we analyze the inherent limitations of current LLMs to pinpoint the structural causes of their suboptimal performance. Finally, we introduce **Sep**arate **Seq**uence (**SepSeq**), a training-free, plug-and-play framework designed to optimize performance via strategic separator insertion.

3.1 Problem Definition

We study long numerical sequence processing tasks. Given a numerical sequence of length n

$$S = [x_1, x_2, \ldots, x_n], \tag{1}$$

where each x_i is a numerical value, and a natural-language query q, the goal is to generate an answer

$$y = f_\theta(q, S). \tag{2}$$

The query may require counting, comparison, filtering, aggregation, trend identification, or other reasoning operations over the sequence. These tasks are challenging for three reasons. First, they are *sequence-complete*: the answer may depend on any element in the sequence, so omitting or corrupting even a small part of the input can change the result (Liu et al., 2024; Dziri et al., 2023). Second, they require *position-sensitive retrieval*: the model must correctly access specific values or local spans within a long sequence rather than rely on coarse semantic summaries. Third, they are *precision-critical*: many tasks involve exact counting or comparison, where small upstream errors can propagate directly into the final answer.

For example, given a stock-price sequence, the query "Excluding non-trading days, how many times did the opening price rise for three or more consecutive days?" requires the model to preserve the sequence faithfully, identify valid local patterns, and aggregate them under explicit constraints. This is not merely a language-understanding problem; it is a precision-critical interaction between natural-language instructions and long numerical input.

3.2 The Bottleneck of Long Numerical Sequences

Long numerical sequences are especially difficult for LLMs because they provide little semantic redundancy. In ordinary text, approximate semantic matching may still preserve the gist of the input. In contrast, numerical sequence tasks often require exact access to individual values or short local spans, and neighboring numbers can be superficially similar while playing very different roles in the final computation. As a result, small errors in attending to, preserving, or referencing the sequence can directly cause answer errors.

One may hope to address this challenge through tool use (Qu et al., 2025), for example, by asking the model to write code or invoke an external solver. However, this does not eliminate the core difficulty. Before any external computation can help, the model must still preserve the input sequence well enough to transcribe, reference, or pass it correctly to the tool. In long numerical inputs, this upstream requirement is itself fragile: if the model drops, alters, or misaligns even a small portion of the sequence, the subsequent tool execution can no longer be trusted. Therefore, tool-assisted reasoning is complementary to, but does not remove, the need for accurate long-sequence input processing.

Existing empirical studies (Dong et al., 2025; Pimentel et al., 2025; Yao et al., 2025; Barbero et al., 2024) highlight the susceptibility of LLMs to errors when transcribing long sequences. To quantify this, we evaluate the LLMs' ability to transcribe numerical sequences verbatim. As detailed in Appendix A, we observe a drastic performance collapse: for sequences exceeding 256 floating-point numbers, the success rate plummets to 14%. This inability to preserve sequence integrity confirms that the bottleneck is intrinsic to the model's processing capacity, effectively rendering tool-based solutions ineffec-

tive.

We attribute this bottleneck to attention dispersion in the standard Softmax attention mechanism:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_k}$ denote the query, key, and value matrices, respectively, with N representing the sequence length. The Softmax function normalizes the attention scores to ensure the weights sum to 1: $\alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^{N} \exp(s_j)}$. Here, s_i denotes the scaled dot-product logit, specifically the element of $\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}$ corresponding to the alignment between the query and the i-th key.

As the sequence length increases, the normalization denominator accumulates contributions from many irrelevant or weakly relevant tokens. For long numerical inputs, where tokens are numerous and often locally similar, this effect dilutes the attention allocated to the truly relevant values. Consequently, the model's attention becomes less focused and more diffuse, making it harder to retrieve the exact numerical evidence needed for downstream reasoning. This phenomenon, illustrated in Figure 1(B), often termed attention dispersion (Jiang et al., 2025; Ahmed et al., 2025; Zsámboki et al., 2025), causes the model to lose focus on critical context, leading to the failures observed in Figure1(A).

3.3 SepSeq: Separate Sequence

Inspired by Chen et al. (2024), 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 SepSeq that guides LLMs to focus attention on local segments rather than the global sequence by artificially introducing specific separators into sequences.

Let $\mathcal{S} = [x_1, \ldots, x_n]$ be a numerical sequence. In the vanilla baseline, a standard delimiter token τ_{sp} (*e.g.,* whitespace, comma) is inserted between adjacent numbers to distinguish them. The proposed SepSeq method imposes structure by replacing τ_{sp} with a specific separator token τ_{sep} (*e.g.,* '\n') at every k-th interval. We formalize this as:

$$\begin{aligned}\text{Vanilla}(\mathcal{S}) &= x_1 \oplus \tau_{sp} \oplus x_2 \oplus \tau_{sp} \oplus \cdots \oplus x_n, \\ \text{SepSeq}(\mathcal{S}, k) &= \underbrace{x_1 \oplus \tau_{sp} \oplus \cdots \oplus x_k}_{\text{Segment 1}} \oplus \tau_{sep} \\ &\quad \oplus \underbrace{x_{k+1} \oplus \tau_{sp} \oplus \cdots \oplus x_{2k}}_{\text{Segment 2}} \oplus \tau_{sep} \oplus \cdots\end{aligned} \tag{4}$$

where $\oplus$ denotes concatenation and k is the segment size. Crucially, this substitution strategy ensures that SepSeq maintains the identical token length as the vanilla baseline, avoiding any overhead in sequence length. We mechanistically analyze how separator tokens alter the attention through the following assumption and theorem.

Assumption 1 (Separator Attention)**.** *Drawing upon observations from Chen et al. (2024), we posit that the separator token τ_{sep}* ***attracts significantly more attention*** *from the subsequent context compared to standard delimiters τ_{sp}. Formally, for a token x_i appearing after the separator, we assume*

$$A[i, \tau_{sep}] > A[i, \tau_{sp}], \tag{5}$$

where $A[i, t]$ denotes the normalized attention weight from position i to token t.

To empirically verify this, we analyze the attention affinity directed from subsequent context towards separator tokens versus standard delimiters. Specifically, we randomly sampled prompts to generate numerical sequences under both Vanilla and SepSeq formats. We then computed the mean attention scores and variances across 10 independent trials, as detailed in Appendix C and visualized in Figure 2(A,B). The results reveal that separator token τ_{sep} consistently elicits significantly higher attention than standard delimiter τ_{sp} at both the layer and head levels, thereby providing strong empirical evidence corroborating Assumption 1.

Theorem 2 (Cross-Segment Attention Suppression)**.** *Under Assumption 1, the separator token τ_{sep} suppresses the attention allocated to tokens in the preceding segment. Formally, for a token x_i appearing after τ_{sep} and a token x_j appearing before τ_{sep},*

$$A_{SepSeq}[i, j] < A_{Vanilla}[i, j]. \tag{6}$$

We analyze the mechanistic role of separator tokens in modulating cross-segment attention patterns. For vanilla method, given a sequence

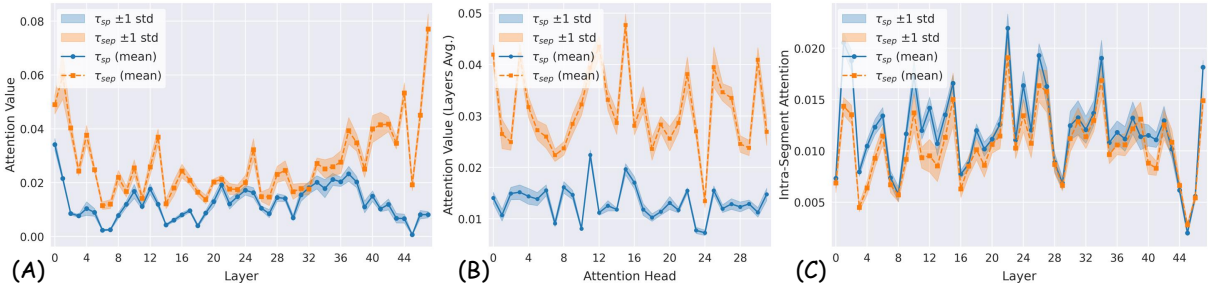

Figure 2: Visualization of the Separator Attention effect. (A-B) The separator τ_{sep} serves as a prominent attention sink, demonstrating consistently higher attention scores than τ_{sp} across both (A) model layers and (B) attention heads. (C) This concentrated attention on τ_{sep} results in a substantial reduction of cross-segment attention weights, thereby localizing the attention scope relative to the vanilla τ_{sp}.

$\mathcal{S}_{vanilla} = [\ldots, x_j, \tau_{sp}, x_i, \ldots]$, the attention score between positions i and j is computed using query vector $\mathbf{Q}_i$, key vector $\mathbf{K}_j$, and key dimension d_k:

$$A_{\text{vanilla}}[i,j] = \frac{\exp\left(\mathbf{Q}_i \cdot \mathbf{K}_j^T / \sqrt{d_k}\right)}{\sum_{l \le i} \exp\left(\mathbf{Q}_i \cdot \mathbf{K}_l^T / \sqrt{d_k}\right)}. \quad (7)$$

In our SepSeq framework, with the sequence $\mathcal{S}_{SepSeq} = [\ldots, x_j, \tau_{sep}, x_i, \ldots]$, the attention calculation remains structurally identical:

$$A_{\text{SepSeq}}[i,j] = \frac{\exp\left(\mathbf{Q}_i \cdot \mathbf{K}_j^T / \sqrt{d_k}\right)}{\sum_{l \le i} \exp\left(\mathbf{Q}_i \cdot \mathbf{K}_l^T / \sqrt{d_k}\right)}. \quad (8)$$

However, strictly distinguishable behaviors emerge in the denominator. As illustrated in Figure 2, τ_{sep} functions as a prominent attention sink, eliciting significantly higher attention scores than standard delimiters τ_{sp}. Specifically, for a token x_i succeeding τ_{sep}, the term $\mathbf{Q}_i \cdot \mathbf{K}_{sep}^T$ is substantial, whereas the baseline counterpart $\mathbf{Q}_i \cdot \mathbf{K}_{sp}^T$ is negligible. This dominance inflates the denominator of the Softmax operation. Consequently, given that the denominator term $Z^{\text{SepSeq}} > Z^{\text{Vanilla}}$, the attention assigned to any distant cross-segment token x_j is asymptotically attenuated:

$$\frac{A_{\text{SepSeq}}[x_i, x_j]}{A_{\text{Vanilla}}[x_i, x_j]} < 1. \quad (9)$$

This theoretical derivation is empirically corroborated in Figure 2(C) (see Appendix C for details).

4 Experiments

To evaluate our approach, we structure our experiments around three core research questions (RQs) regarding performance, robustness, and efficiency:

RQ1 – Effectiveness and Generalization. Does our proposed method consistently enhance model performance across diverse tasks and architectures, demonstrating strong generalizability?

RQ2 – Robustness and Sensitivity. Which factors modulate the effectiveness of our method, and how can it be optimally configured for different scenarios?

RQ3 – Efficiency and Cost. Does our method reduce token consumption during inference?

The remainder of this section is organized as follows: We first describe the experimental setup, including datasets, evaluation metrics, and baselines. We then provide a detailed analysis of the experimental results corresponding to each RQ.

4.1 Experimental Settings

4.1.1 Dataset

We design two datasets to conduct an in-depth investigation of LLMs' capabilities for processing long numerical sequences: a synthetic dataset $\mathcal{D}_{\text{syn}}$ and a real dataset $\mathcal{D}_{\text{real}}$.

For $\mathcal{D}_{\text{syn}}$, we construct sequences of varying lengths comprising both integer and floating-point numbers. We categorize these sequences into four length intervals: S (short) for sequences containing [2, 32] (*i.e.,* $2 \le n \le 32$) numbers, M (medium) for (32, 128] numbers, L (large) for (128, 256] numbers, and XL (extra-large) for (256, 512] numbers. We formulate six distinct task types: (1) *max-int*, which requires identifying the index of the maximum integer in an integer sequence; (2) *min-int*, which locates the index of the minimum integer; (3) *max-float* and (4) *min-float*, which perform analogous operations on floating-point sequences; (5) *indexing*, which determines the position of the last occurrence of 1 in a binary sequence; and (6) *counting*, which counts the total number of 1s in a binary sequence. Each task comprises 200 samples, with 50 samples distributed across each of the four

length categories.

Building upon the prior work of Li et al. (2025), we construct $\mathcal{D}_{\text{real}}$ to assess model performance on practical numerical reasoning tasks. This dataset comprises four distinct categories, each containing 200 samples. The categories include: (1) *number-string*, which involves counting numerals in alphanumeric sequences; (2) *number-list*, requiring logical reasoning over numerical sequences; and (3-4) *stock* and *weather*, both constructed from real-world datasets with human-generated questions. Notably, any failure to process a single value in these tasks inevitably results in an incorrect final answer, making them particularly challenging. See Appendix B for details (*e.g.,* examples, statistical information).

4.1.2 Evaluation Metrics

To comprehensively evaluate our proposed method, we assess both performance and robustness using the following metrics:

Accuracy (Acc). This metric serves as the primary indicator of performance, quantifying the ratio of correct predictions to the total number of test instances. Specifically, let N_{correct} be the count of correct answers and N be the total number of questions; Accuracy is defined as:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N}. \tag{10}$$

Answer Rate (AR). Answer Rate measures reliability as the ratio of valid (non-null) responses to total inputs. Let N be the total queries and N_{valid} the count of successful generations, defined as:

$$\text{AR} = \frac{N_{\text{valid}}}{N}. \tag{11}$$

4.1.3 Base Models

We conduct a comprehensive evaluation across 9 LLMs, representing diverse architectures, parameter scales, and training paradigms from both open-source and proprietary domains. **Open-source models:** Our selection includes the Qwen3 family (Yang et al., 2025), spanning 0.6B to 30B parameters with both dense and Mixture-of-Experts architectures (Fedus et al., 2022; Zhou et al., 2023), available in instruct and reasoning modes, alongside the QwQ-32B model. We also evaluate the DeepSeek series (Guo et al., 2025), including the recent DeepSeek-R1 and DeepSeek-V3 variants, which are known for their strong reasoning capabilities. **Proprietary models:** We assess Claude-3.7-Sonnet from Anthropic (Anthropic, 2025). Additionally, we evaluate Google's Gemini-2.5-Pro (Gemini-Team, 2025), which showcases multimodal understanding capabilities, and two variants from OpenAI's GPT-4 series (Achiam et al., 2023): GPT-4.1 and GPT-4o. See Section E for a detailed version of the models.

4.1.4 Baselines

To rigorously evaluate the efficacy of our framework, we compare it against established inference paradigms. **Vanilla:** As proposed by Barbero et al. (2024), we utilize standard delimiters (*e.g.,* comma) to separate numerical entries. This serves as a fundamental baseline, providing basic visual separation to enhance the representational distinctiveness of adjacent numbers in long sequences. **Reasoning-Enhanced Inference:** We adopt Chain-of-Thought (CoT) prompting (Wei et al., 2023) to activate the model's sequential reasoning. By explicitly prompting for a step-by-step derivation, this method enforces the decomposition of the problem into intermediate deductive steps. **In-Context Learning:** Leveraging the in-context learning (ICL) capability of LLMs (Yu et al., 2022), this baseline provides a single exemplar within the prompt context. This setup tests the model's ability to induce the task format and logical pattern from sparse supervision.

4.2 Experimental Results

We present a comprehensive evaluation designed to address the proposed Research Questions (RQs). Each RQ is investigated through multiple complementary analytical lenses, providing rigorous empirical evidence to substantiate our claims.

4.2.1 Effectiveness and Generalization (RQ1)

Across tasks. We evaluate our method against baselines on six synthetic and four real tasks, reporting Accuracy and Answer Rate. Analysis in Table 1 reveals that standard LLMs struggle with long numerical sequences. Notably, methods like CoT and ICL prompting prove detrimental, dropping average accuracy from 51.6% to 49.0% and 48.9%, respectively. This strongly suggests that LLM's failures stem not from an insufficient reasoning ability but from a fundamental inability to parse and manage numerical sequences properly.

Conversely, SepSeq achieves a 35.6% relative accuracy gain (averaging 69.9%) over the baseline. The improvement is particularly pronounced in real-world tasks, with surges of +100.0% on Stock

Table 1: The average answer rate and accuracy for each task over 10 independent runs, reported as mean ± standard deviation. **Bold** indicates the best average performance, while underlined values denote the best average performance excluding our method. "Incr." indicates the percentage improvement of SepSeq over the strongest non-SepSeq baseline. Green and red indicate improvement and degradation in performance, respectively.

Task	Answer Rate (%, ↑)				Accuracy (%, ↑)			
	Vanilla	CoT	ICL	SepSeq (Incr.)	Vanilla	CoT	ICL	SepSeq (Incr.)
counting	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	42.7±2.7	41.2±3.0	37.9±3.2	**76.7±1.2** (+79.6%)
indexing	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	38.8±3.1	33.4±3.4	34.9±2.9	**86.6±0.8** (+123.0%)
max-float	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	63.9±1.9	60.5±2.1	63.3±1.7	**81.0±1.1** (+26.8%)
max-int	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	75.3±1.1	68.8±1.7	71.0±1.4	**92.4±0.4** (+22.7%)
min-float	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	63.1±1.7	60.2±2.2	60.4±1.9	**79.3±1.0** (+25.7%)
min-int	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	74.3±1.4	68.1±1.5	66.5±1.8	**93.0±0.3** (+25.1%)
number-string	96.6±0.2	96.3±0.3	96.3±0.1	**99.0±0.1** (+2.5%)	81.6±1.0	81.0±0.8	**81.7±1.1**	**81.7±0.9** (+0.0%)
number-list	78.2±1.2	77.6±0.9	73.4±1.4	**79.1±1.1** (+1.1%)	36.3±3.3	35.4±3.1	32.6±3.5	**36.7±3.2** (+0.9%)
stock	64.1±1.7	63.1±1.9	63.3±1.6	**73.6±1.4** (+14.7%)	13.2±4.4	13.7±4.2	13.4±4.5	**27.4±3.5** (+100.0%)
weather	66.0±1.6	65.6±1.8	65.9±1.5	**76.8±1.2** (+16.3%)	26.7±3.8	27.2±3.5	27.8±3.9	**44.6±2.8** (+60.9%)
Average	90.5±0.4	90.3±0.6	89.9±0.5	**92.8±0.3** (+2.6%)	51.6±2.5	49.0±2.7	48.9±2.4	**69.9±1.6** (+35.6%)

Table 2: The average answer rate and accuracy of each model across 10 tasks, computed over 10 independent runs and reported as mean ± standard deviation. **Bold** indicates the best average performance, while underlined values denote the best average performance excluding our method. "Incr." indicates the percentage improvement of SepSeq over the strongest non-SepSeq baseline. Green and red indicate improvement and degradation in performance, respectively.

Model	Answer Rate (%, ↑)				Accuracy (%, ↑)			
	Vanilla	CoT	ICL	SepSeq (Incr.)	Vanilla	CoT	ICL	SepSeq (Incr.)
Qwen3-8B	79.4±1.1	79.5±0.9	75.2±1.3	**87.3±0.6** (+9.8%)	45.5±2.8	40.3±2.9	38.5±3.2	**69.6±1.6** (+53.0%)
Qwen3-30B-A3B	80.6±0.9	80.9±1.1	81.0±0.8	**90.2±0.5** (+11.4%)	54.3±2.4	52.7±2.2	52.3±2.5	**80.7±1.1** (+48.6%)
QwQ-32B	72.7±1.5	72.2±1.3	72.7±1.2	**75.6±1.1** (+4.0%)	34.2±3.4	28.9±3.7	28.5±3.5	**57.8±2.0** (+69.0%)
DeepSeek-V3	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0 (+0.0%)	45.7±2.6	44.5±2.9	45.2±2.7	**50.9±2.5** (+11.4%)
DeepSeek-R1	**99.9±0.1**	**99.9±0.0**	99.8±0.1	**99.9±0.0** (+0.0%)	61.1±1.8	56.6±2.3	57.9±2.0	**70.5±1.4** (+15.4%)
Claude-3.7-Sonnet	99.8±0.1	99.9±0.0	**100.0±0.0**	99.9±0.1 (-0.1%)	57.3±2.2	57.6±2.0	54.2±2.4	**79.1±1.1** (+37.3%)
Gemini-2.5-Pro	83.4±0.9	81.8±1.0	82.0±0.8	**84.6±0.7** (+1.4%)	58.9±2.2	48.0±2.7	56.9±2.0	**79.1±1.0** (+34.3%)
GPT-4.1	99.6±0.1	99.6±0.0	**99.8±0.1**	99.5±0.1 (-0.1%)	61.0±1.9	60.0±2.1	60.0±1.8	**82.7±0.9** (+35.6%)
GPT-4o	**99.0±0.1**	98.5±0.2	98.6±0.1	98.6±0.1 (-0.4%)	46.4±2.6	51.5±2.5	44.5±2.9	**59.1±2.1** (+14.8%)
Average	90.5±0.5	90.3±0.4	89.9±0.6	**92.8±0.3** (+2.6%)	51.6±2.4	48.9±2.7	48.7±2.5	**68.9±1.5** (+35.6%)

and +60.9% on Weather datasets. Furthermore, SepSeq improves inference reliability (+2.6% answer rate), demonstrating that simple input restructuring effectively rectifies sequence understanding failures without model retraining.

Across models. We evaluate 9 diverse high-performance LLMs, presenting model-wise performance gains in Table 2 to validate the general applicability of our approach. Our analysis demonstrates that SepSeq yields robust improvements averaging +35.6% accuracy, irrespective of model family or architectural design. **Model Families.** SepSeq proves effective across a wide spectrum of model lineages. It delivers consistent gains for proprietary state-of-the-art models, including GPT-4.1 (+35.6%), Claude-3.7-Sonnet (+37.3%), and Gemini-2.5-Pro (+34.3%). Similarly, open-weights models from the Qwen and DeepSeek families exhibit comparable or even larger surges (e.g., QwQ-32B +69.0%), confirming that the method's efficacy is not tied to a specific training recipe or alignment strategy. **Architectures (Dense vs. MoE).** Crucially, the improvements extend across distinct architectural paradigms. For standard Dense models like Qwen3-8B, SepSeq unlocks a massive +53.0% accuracy boost. The framework is equally potent for MoE architectures, such as Qwen3-30B-A3B, improving accuracy by 48.6%.

4.3 Robustness and Sensitivity (RQ2)

Separate interval. An analysis of the separator interval (k) reveals that our method exhibits remarkable stability once a minimal segmentation frequency is met. As shown in Figure 3(A), performance gains saturate rapidly: accuracy plateaus

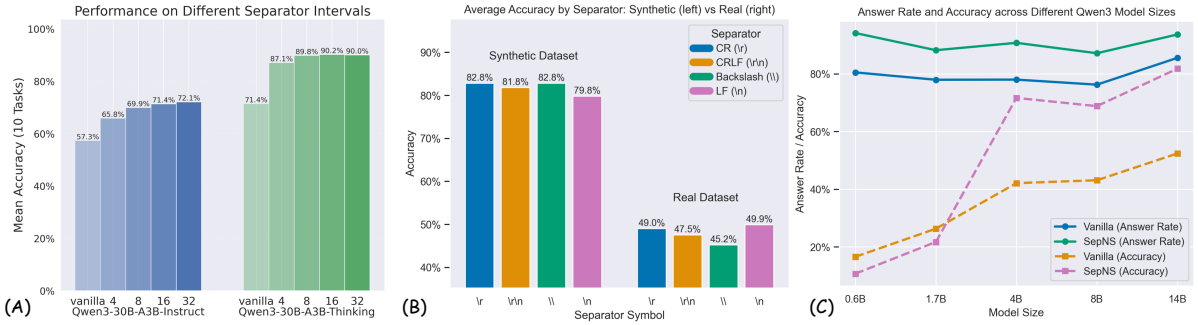

Figure 3: (A): Average Accuracy across different separate intervals. (B): Average Accuracy across different separator symbols. (C): Average Answer Rate and Accuracy across different Qwen3 model sizes: 0.6B to 14B.

significantly for intervals $k \geq 8$, showing negligible variance for larger windows ($k = 16, 32$). Specifically, the Thinking model maintains a consistent near-optimal accuracy of $\sim 90\%$ across $k \in \{8, 16, 32\}$, with deviations of less than 0.4%. Similarly, the Instruct model stabilizes in the 70-72% range beyond this threshold. This observation suggests that SepSeq is hyperparameter-robust, providing substantial benefits without necessitating precise tuning of the segment length. We use $k = 16$ in our main experiments. Based on these results, we set $k = 16$ for our main experiments.

Separator symbol. We investigate separator symbol impact on performance using Qwen3-30B-A3B-Instruct-2507 across 10 tasks with four separator types: Carriage Return (CR, `\r`), Carriage Return Line Feed (CRLF, `\r\n`), Backslash (`\\`), and Line Feed (LF, `\n`). Tasks are categorized into basic numerical processing ($\mathcal{D}_{\text{syn}}$) and complex applications ($\mathcal{D}_{\text{real}}$).

Figure 3(B) reveals systematic performance differentiation across separator types varying with task complexity. For basic tasks $\mathcal{D}_{\text{syn}}$, all separators showed modest accuracy variations (79.8%–82.8%), with unconventional separators CR and Backslash achieving optimal accuracy (82.8% each), potentially due to novelty requiring enhanced attention. However, inversion emerged in complex scenarios: while CR and Backslash excel in basic tasks, LF demonstrated superior performance (49.9% vs. Backslash's 45.2%) in complex tasks. This reversal suggests fundamental processing strategy shifts across complexity levels. These patterns evidence sophisticated cognitive resource allocation (Sweller, 1988; Sweller et al., 2011) in LLMs. Under low cognitive load, models allocate additional resources to separator adaptation, where novel separators benefit from enhanced attention.

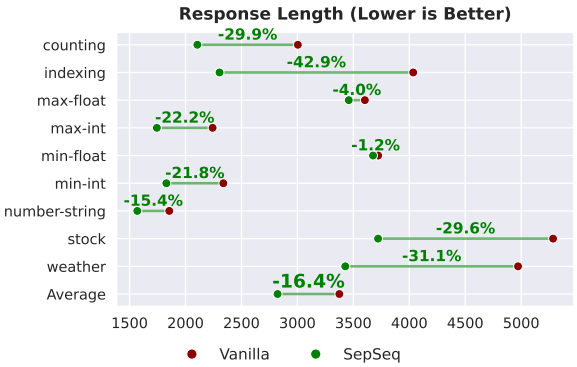

Figure 4: Comparison between Vanilla and SepSeq across tasks. The left panel shows that SepSeq significantly reduces token consumption during inference (average -16.4%).

Under high cognitive load, models prioritize core semantic processing, favoring minimal-overhead separators. LF's superior complex performance reflects the prevalence of training data and processing efficiency, enabling more resources for task-specific reasoning over format adaptation.

Model size. We evaluated Qwen3 models at various parameter scales (0.6B, 1.7B, 4B, 8B, and 14B). The results, shown in Figure 3(C), demonstrate that the effectiveness of SepSeq is scale-dependent. For smaller models (0.6B), SepSeq underperforms the vanilla baseline, suggesting a minimum capacity requirement for effective separator interpretation. A critical inflection point occurs at the 4B parameter level, where SepSeq begins to yield performance gains. Beyond this scale, the performance gap widens substantially, reaching over 80% accuracy on the 14B models.

Reasoning vs. Instruction models. While SepSeq benefits different architectures, Figure 3(A) elicits a stronger response from reasoning-enhanced models. First, the Thinking model's

Prompting	Input Formatting	
	Standard	SepSeq
Vanilla	53.25	70.35
PoT	78.90	**84.45**

Table 3: Average results for PoT across 10 tasks (200 samples/task).

baseline without separators (71.4%) matches the Instruct model's saturated peak (72.1%), indicating an intrinsic advantage in handling long contexts. Second, when augmented with SepSeq, the Thinking model achieves a massive leap to 90.2% accuracy, breaking the performance ceiling that constrains the Instruct model. In contrast, the Instruct model, while improving from 57.3% to 72.1%. This disparity indicates that SepSeq acts as a structural multiplier for reasoning: it provides the necessary attention landmarks that reasoning models are uniquely equipped to exploit, unlocking superior numerical processing capabilities.

Comparison with Program-of-Thought. To clarify whether SepSeq replaces or complements tool-assisted reasoning, we compare it against a Program-of-Thought (PoT) baseline in a compact 2×2 setting: *Vanilla* vs. *PoT*, each evaluated with and without SepSeq formatting. SepSeq is applied only to the input numerical sequence, while the PoT prompting and execution pipeline remain unchanged. As shown in Table 3, across 10 tasks (200 samples per task), SepSeq consistently improves performance in both settings. Under standard prompting, accuracy increases from 53.25% to 70.35%; under PoT, it further improves from 78.90% to 84.45%. The gain persists even when combined with structured code-based reasoning, with PoT+SepSeq achieving the best overall result.

These findings indicate that SepSeq is complementary to PoT rather than redundant with it. While PoT primarily strengthens downstream computation, SepSeq improves the upstream processing of long numerical inputs, making the two approaches naturally compatible.

4.4 Efficiency and Cost (RQ3)

Token Consumption. We analyze the computational efficiency of SepSeq regarding token usage. One might worry that inserting separator tokens increases the total sequence length, thereby increasing inference cost. However, Figure 4 presents a favorable outcome: SepSeq consistently reduces both response length and total token consumption across most tasks.

As shown in the chart, the *Response Length* exhibits a drastic reduction. The average response length drops from 3,375 tokens (Vanilla) to 2,823 tokens (SepSeq), representing a 16.4% reduction. In specific tasks like *Indexing* and *Stock*, the reduction is even more pronounced, reaching approximately 42.9% and 29.6%, respectively. This efficiency stems from SepSeq's ability to mitigate the "repetition loops" and hallucinated gibberish often observed in vanilla LLMs when they lose track of long sequences. By maintaining focused attention, SepSeq generates concise, precise answers without redundant tokens. This confirms that SepSeq is not only an accuracy-enhancing framework but also a cost-effective solution that reduces inference latency and API costs.

5 Conclusion

In this work, we identify and address a fundamental limitation of LLMs: attentional dispersion over long numerical sequences, which precipitates severe performance degradation in precision-critical tasks. To mitigate this, we introduce SepSeq, a training-free framework that strategically inserts separators to partition sequences into manageable segments. Through evaluation across 9 state-of-the-art models and 10 diverse tasks, SepSeq achieves a substantial 35.6% average accuracy improvement. Crucially, this gain comes without additional training and reduces net token consumption. Our analysis reveals that separators induce localized attention patterns, acting as anchors that transform dispersed attention into focused segment processing. This demonstrates that simple input restructuring serves as a powerful mechanism to unlock LLM numerical capabilities, offering a scalable and efficient solution for real-world applications.

Limitations

While SepSeq demonstrates significant improvements in processing long numerical sequences, we identify the limitation in our current study:

Scale Dependence. Our evaluation on the Qwen3 family reveals that SepSeq's effectiveness is closely tied to model scale. Specifically, models with fewer parameters (*e.g.,* 0.6B) exhibit performance degradation compared to the vanilla baseline, likely due to their limited capacity to interpret the implicit structural guidance provided by sepa-

rators. The performance gains only become robust significantly beyond the 4B parameter threshold, suggesting that our method requires a certain level of emergent cognitive capability to be effective.

Ethical Consideration

The proposed SepSeq framework is a training-free inference technique that modifies input formatting without altering model parameters or requiring additional data collection. Our experiments utilize publicly available models and synthetic datasets, with no involvement of human subjects, collection of personal data, or privacy risks. The method enhances model accuracy in numerical processing tasks without introducing harmful capabilities or creating potential for misuse. All experimental evaluations were conducted using established benchmarks and standard evaluation protocols. The research makes a positive contribution to the field by addressing fundamental limitations in LLM numerical processing capabilities, with potential benefits for applications that require precise numerical computation.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (62302321) and the Fundamental Research Funds for the Central Universities (WK2100250065).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint*, abs/2303.08774.

Murtadha H. M. Ahmed, Wen Bo, and Yunfeng Liu. 2025. MateICL: Mitigating attention dispersion in large-scale in-context learning. *arXiv preprint*, abs/2505.01110.

Anthropic. 2025. System card: Claude opus 4 & claude sonnet 4.

Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João G.M. Araújo, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. 2024. Transformers need glasses! Information oversquashing in language tasks. In *Advances in Neural Information Processing Systems 37*, pages 98111–98142, Vancouver, Canada.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. 2024. SepLLM: Accelerate large language models by compressing one segment into one separator. In *Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2024. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA.

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

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. ReTool: Reinforcement learning for strategic tool use in LLMs. *arXiv preprint*, abs/2504.11536.

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

Gemini-Team. 2025. Gemini 2.5 pro model card. https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf.

GLM-4.5-Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei,

and 152 others. 2025. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv preprint*, abs/2508.06471.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint*, abs/2501.12948.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, New Orleans, LA.

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: what's the real context size of your long-context language models? In *Proceedings of the 2024 Conference on Language Modeling*, Philadelphia, PA.

Xinyan Jiang, Hang Ye, Yongxin Zhu, Xiaoying Zheng, Zikang Chen, and Jun Gong. 2025. HICD: Hallucination-inducing via attention dispersion for contrastive decoding to mitigate hallucinations in large language models. In *Findings of the Association for Computational Linguistics*, pages 7764–7786, Vienna, Austria.

Kimi-Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, and 150 others. 2025. Kimi K2: Open agentic intelligence. *arXiv preprint*, abs/2507.20534.

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

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

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

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

Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. 2023. Scaling laws of rope-based extrapolation. *arXiv preprint*, abs/2310.05209.

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

Ken M. Nakanishi. 2025. Scalable-softmax is superior for attention. *arXiv preprint*, abs/2501.19399.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint*, abs/2309.00071.

Tiago Pimentel and 1 others. 2025. Repetitions are not all alike: Distinct mechanisms sustain repetition in language models. *arXiv preprint*, abs/2504.01100.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint*, abs/2504.13958.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8):198343.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA.

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

John Sweller, Paul Ayres, and Slava Kalyuga. 2011. *Cognitive Load Theory*. Springer, New York.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint*, abs/1706.03762.

Petar Velickovic, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. 2024. Softmax is not enough (for sharp size generalisation). *arXiv preprint*, abs/2410.01104.

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

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *Proceedings of the the 12th International Conference on Learning Representations*, Vienna, Austria.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint*, abs/2505.09388.

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

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

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

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Manoj Awalgaonkar, Rithesh R. N., Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, and 2 others. 2025. xlam: A family of large action models to empower AI agent systems. In *NAACL (Long Papers)*, pages 11583–11597. Association for Computational Linguistics.

Yiming Zhang, Shi Li, and Yang Liu. 2022. On the robustness of chinese text processing pipelines. *Findings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 3548–3560.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jianfeng Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint*, abs/2305.14705.

Pál Zsámboki, Benjamin Levi, David Ansel Josef Smith, Mitansh Kagalwala, Arlington Kell, Samuel Liechty, and Cong Wang. 2025. Learning what's missing: Attention dispersion and EMA stabilization in length generalization. *arXiv preprint*, abs/2510.08341.

A The Long-Context Copying Bottleneck

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

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

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

These findings highlight significant architectural or attentional limitations in current LLMs for tasks demanding precise, long-sequence replication. Such failures may stem from compounding errors in the attention mechanism, effective context window constraints, or biases in the training data.

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

Size	Total	Qwen3-30B-A3B-Instruct Model		Qwen3-30B-A3B-Thinking-2507	
		# Correct	Accuracy	# Correct	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%
Overall	250	157	62.80%	102	40.80%

The inferior performance of the "Thinking" variant is particularly noteworthy. It suggests that for rote, mechanical tasks that do not require reasoning, the cognitive overhead or architectural modifications intended to facilitate complex thought may act as a source of noise, thereby degrading performance on simple memorization and reproduction.

B Dataset Details

This appendix provides detailed descriptions of the datasets used to evaluate LLMs' long numerical sequence processing capabilities.

B.1 Synthetic Dataset

The synthetic dataset D_{syn} consists of 1,200 samples across six task types, each containing 200 samples. The sequences are categorized into four length intervals:

- **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)

B.1.1 Task Types

max-int: Identify the index (0-based) of the maximum integer in an integer sequence. Example:

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

The maximum value is 5 at index 7.

min-int: Identify the index (0-based) of the minimum integer in an integer sequence. Example:

```
{
  "task_type": "min_int",
  "answer": "19",
  "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, 6, 9, 6]
}
```

The minimum value is 2 at index 19.

max-float: Identify the index (0-based) of the maximum floating-point number in a sequence. Similar to max-int but with floating-point numbers.

min-float: Identify the index (0-based) of the minimum floating-point number in a sequence. Similar to min-int but with floating-point numbers.

indexing: Determine the position of the last occurrence of 1 in a binary sequence. Example:

```
{
  "task_type": "indexing",
  "answer": "8",
  "ts": [1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
}
```

The last occurrence of 1 is at index 8.

counting: Count the total number of 1s in a binary sequence. Example:

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

There are 4 occurrences of 1 in the sequence.

B.2 Real Dataset

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

B.2.1 Task Categories

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

```
{
  "question": "How many numbers are there
      in the string? Note that a
      sequence like 'a243b' counts as a
      single number.",
  "struct_data":
      "effV2xM8hF5vcNgl8xrTCmbD6sEM38ti",
  "answer": 11
}
```

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

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

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

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

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

```
{
  "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, "...":
        "..."},
    {"date": "2024-11-13",
        "temperature_2m": 4.92,
        "precipitation": 0.0, "...":
        "..."},
    {"date": "2024-11-14",
        "temperature_2m": 5.45,
        "precipitation": 0.0, "...":
        "..."},
    "..."
  ],
  "answer": "B"
}
```

Listing 2: Weather example (truncated)

This task involves analyzing real-world meteorological time series data with questions about temperature patterns, precipitation, and temporal trends.

B.3 Dataset Statistics

Table 5 shows the dataset statistics.

C Separator Attention Effect

C.1 Separator Attention

Experimental Setup. To quantify the attention distribution mechanism, we conducted a controlled experiment using the Qwen3-30B-A3B-Thinking model. We generated 10 independent trials of random numerical sequences, each consisting of 20 integers (range 0-9). The sequences were formatted using two configurations: **Vanilla:** Uses standard spaces (τ_{sp}) as delimiters; **SepSeq:** Inserts newline separators (τ_{sep}) every 8 tokens.

Metric Definition. We extracted the attention weights from tokens in subsequent segments (specifically indices 9-16 and 17-20) directed towards the separator tokens located at the segment boundaries (indices corresponding to the 1st and 2nd separators). Figure 2(A) reports the attention scores averaged across all heads for each layer, with shaded regions representing ± 1 standard deviation across the 10 sequences. Figure 2(B) reports the attention scores for each attention head index

Table 5: Comprehensive Token Length Statistics. Comparison between Question Only and Full Prompt across six statistical dimensions: Minimum, Quartiles (25%, 50%, 75%), Mean, and Maximum. All tasks consist of 200 samples.

Task Type	Question Only (Tokens)						Full Prompt (Tokens)						#Samples
	Min	25%	50%	Mean	75%	Max	Min	25%	50%	Mean	75%	Max	
max_int	56	56	56	56.0	56	56	88	153	341	411.9	594	1070	200
min_int	56	56	56	56.0	56	56	88	149	341	410.3	594	1104	200
max_float	56	56	56	56.0	56	56	106	318	957	1252.8	1867	3613	200
min_float	56	56	56	56.0	56	56	99	334	968	1232.3	1869	3562	200
counting	24	24	24	24.0	24	24	56	117	309	392.9	565	1076	200
indexing	43	43	43	43.0	43	43	75	138	328	408.8	588	1085	200
weather	66	92	134	119.0	140	145	9630	9817	9853	9859.3	9910	10127	200
stock	67	86	131	117.8	138	151	11024	11493	11674	11746.9	11992	12521	200
number_list	55	83	100	116.1	162	194	640	772	5079	3916.1	5124	5195	200
mixed_number_string	27	27	27	27.0	27	27	128	135	137	136.7	139	141	200

(0-31), averaged across all layers and sequences, to illustrate head-specific behaviors.

C.2 Cross-Segment Attention Suppression

Experimental Setup for Cross-Segment Attention Analysis. To investigate the impact of separator tokens on long-range dependency modeling, we analyzed the inter-segment attention patterns using the same Qwen3-30B-A3B-Thinking model and data generation protocol as the previous experiment (10 independent sequences of length 20). We compared two formatting strategies: **Vanilla:** Standard space-delimited sequence; **SepSeq:** Segmented sequence with newline separators inserted every 8 tokens.

Metric Definition. We defined Cross-Segment Attention as the aggregate attention weight originating from tokens in a current segment and directed towards tokens in all preceding segments. Specifically, we computed the attention flow for two cross-segment scenarios: (1) From tokens in the 2nd segment (indices 9-16) attending to tokens in the 1st segment (indices 1-8); (2)From tokens in the 3rd segment (indices 17-20) attending to tokens in the combined 1st and 2nd segments (indices 1-16).

Visualization. The Figure 2(C) reports the mean cross-segment attention scores averaged across all attention heads and selected token pairs for each model layer. Shaded areas indicate the standard deviation across the 10 independent trials. A lower value in the SepSeq configuration indicates effective suppression of attention to distant, irrelevant context.

D The Use of Large Language Models

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

E Reproducibility statement

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

Code and Data. Our proposed method is thoroughly detailed in Section 3, including algorithmic descriptions and implementation specifics. Complete source code and datasets are available through the anonymous repository at https://github.com/sunjie279/SepSeq, with the accompanying README file providing step-by-step instructions for execution and reproduction of experiments.

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

1. Qwen3-8B: https://huggingface.co/Qwen/Qwen3-8B
2. Qwen3-30B-A3B: https://huggingface.co/Qwen/Qwen3-30B-A3B
3. QwQ-32B: https://huggingface.co/Qwen/QwQ-32B
4. DeepSeek-V3: https://huggingface.co/deepseek-ai/DeepSeek-V3-0324

5. DeepSeek-R1: https://huggingface.co/deepseek-ai/DeepSeek-R1-0528

6. Claude-3.7-Sonnet: https://openrouter.ai/anthropic/claude-3.7-sonnet

7. Gemini-2.5-Pro: https://openrouter.ai/google/gemini-2.5-pro

8. GPT-4.1: https://openrouter.ai/openai/gpt-4.1

9. GPT-4o: https://openrouter.ai/openai/gpt-4o-2024-08-06

For robustness evaluation (RQ2), **separator interval analysis**, we compare instruction and reasoning variants:

1. Qwen3-30B-A3B-Instruct: https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507

2. Qwen3-30B-A3B-Thinking: https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507

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

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

1. Qwen3-0.6B: https://huggingface.co/Qwen/Qwen3-0.6B

2. Qwen3-1.7B: https://huggingface.co/Qwen/Qwen3-1.7B

3. Qwen3-4B: https://huggingface.co/Qwen/Qwen3-4B

4. Qwen3-8B: https://huggingface.co/Qwen/Qwen3-8B

5. Qwen3-14B: https://huggingface.co/Qwen/Qwen3-14B