STREAMINGTHINKER: LARGE LANGUAGE MODELS CAN THINK WHILE READING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have demonstrated remarkable capabilities in chain of thought (CoT) reasoning. However, the current LLM reasoning paradigm initiates thinking only after the entire input is available, which introduces unnecessary latency and weakens attention to earlier information in dynamic scenarios. Inspired by human cognition of thinking while reading, we first design a ***streaming thinking*** paradigm for LLMs, where reasoning unfolds in the order of input and further adjusts its depth once reading is complete. We instantiate this paradigm with *StreamingThinker*, a framework that enables LLMs to think while reading through the integration of streaming CoT generation, streaming-constraint training, and streaming parallel inference. Specifically, StreamingThinker employs streaming reasoning units with quality control for CoT generation, enforces order-preserving reasoning through streaming attention masks and position encoding, and leverages parallel KV caches that decouple input encoding from reasoning generation, thereby ensuring alignment and enabling true concurrency. We evaluate StreamingThinker on the Qwen3 model family across math reasoning, logical reasoning, and context-based QA reasoning tasks. Experimental results show that the StreamingThinker preserves performance comparable to batch thinking, while yielding an 80% reduction in token waiting before the onset of reasoning and a more than 60% reduction in time-level latency for producing the final answer, demonstrating the effectiveness of the streaming paradigm for LLM reasoning.

1 INTRODUCTION

Large language models (LLMs) have shown impressive reasoning capabilities, as exemplified by systems like OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025). Yet, most current approaches follow a ***batch thinking*** paradigm, in which reasoning begins only after the entire input context has been received. This paradigm is problematic in scenarios that demand timely responses or dynamic information processing. First, waiting for the full input before reasoning introduces unnecessary latency. Second, as the input increases, attention to earlier information becomes diluted due to the growing disconnect between reasoning steps and their relevant context (Liu et al., 2024; Levy et al., 2024; Zhang et al., 2025b). This weakens coherence and raises the risk of hallucination. To compensate, LLMs often rely on longer chains of thought (CoT) (Wei et al., 2022; Yeo et al., 2025; Chen et al., 2025b) or repeated self-refinement (Wang et al., 2023; Ling et al., 2023; Madaan et al., 2023) to re-focus, which in turn raise computational costs and inflate token usage.

In contrast, human reasoning often unfolds in an immediate and streaming manner. Research in psychology and cognitive science shows that during reading, humans process incoming information instantaneously, including text decoding, meaning construction, background knowledge activation, integrative reasoning, and actively generating inferences to build a coherent understanding (Kintsch, 1988; Graesser et al., 1994). This *"thinking while reading"* mechanism not only enhances processing efficiency, but also allows reasoning to occur closely alongside the relevant context, minimizing cognitive lag and mitigating the risk of coherence degradation.

To narrow the gap between LLM and human reasoning, we propose a ***streaming thinking*** paradigm for LLMs. Streaming thinking unfolds reasoning steps alongside the input stream, allowing the model to reason while receiving information. Once the full input is available, the model can further

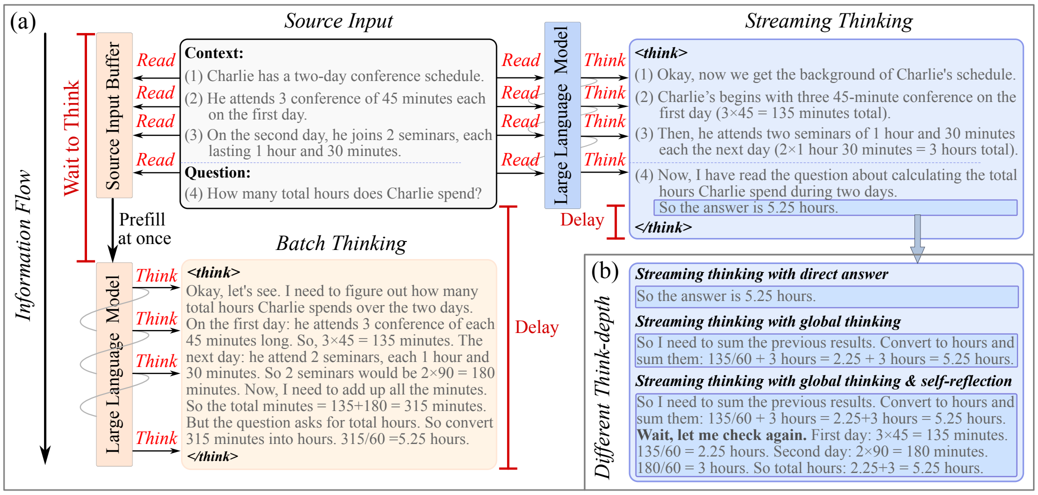

Figure 1: (a) Standard LLM reasoning follows the ***batch thinking*** paradigm, where reasoning begins only after the entire input is received, leading to high latency and imbalanced attention to the input. The proposed ***streaming thinking*** paradigm enables LLMs to think while reading during input reception, substantially reducing latency and maintaining attention aligned with the order of input. (b) Streaming thinking paradigm supports multi-depth reasoning, balancing latency with performance.

refine its reasoning and adjust the depth of its analysis to match task complexity.[1] As illustrated in Figure 1, compared with batch thinking, streaming thinking enables much faster responses while preserving consistency with the order of incoming information.

Accordingly, we propose *StreamingThinker*, a framework that instantiates the streaming thinking paradigm. The StreamingThinker integrates a streaming CoT generation pipeline with training and inference frameworks that adapt LLMs to the streaming paradigm. The generation pipeline inserts boundary tokens to the inputs to define minimal reasoning units, prompting LLMs to generate serialized reasoning segments for each unit that are reconstructed into incremental content, filtered by quality evaluation, with reasoning depth controlled through token intervention (Wu et al., 2025). To support streaming training, StreamingThinker introduces two modifications: a streaming attention mask that restricts each reasoning step to past and current input, and a streaming position encoding that independently indexes input and reasoning tokens from zero to eliminate positional contention, ensuring alignment with associated inputs. For inference, StreamingThinker employs parallel KV caches that decouple input encoding from reasoning generation and merge only during cross-attention, enabling true thinking while reading. We conduct a comprehensive evaluation on Qwen3 model family (Yang et al., 2025), covering diverse tasks such as math reasoning, logical reasoning, and context-based QA reasoning. Experimental results indicate that StreamingThinker achieves reasoning performance on par with batch thinking, yet reduces token-level waiting before reasoning by 80% and overall answer latency by over 60%.

The contributions of this work are fourfold.

- To the best of our knowledge, we are the first to introduce the streaming thinking paradigm for large language model reasoning. This paradigm mirrors human cognitive processes, enabling LLMs to engage in more timely and continuous thinking in dynamic scenarios.
- We propose a streaming CoT generation pipeline for this paradigm. Drawing on the principles of human streaming thinking, it ensures that the reasoning process remains aligned with the sequential order of the input context.
- We provide an adaptation training and inference framework that implements the streaming thinking paradigm, in which training ensures alignment with sequential inputs and inference enables efficient concurrent reasoning.
- Extensive experiments on diverse reasoning tasks show that our method achieves reasoning performance comparable to batch thinking, while markedly reducing waiting latency.

[1] Appendix A discusses the value and potential applications of human-like streaming thinking in practice.

2 Streaming Thinking Paradigm

Paradigm Design Human streaming cognition involves two complementary processes: rapidly generating and updating representations as input arrives, and subsequently performing global integration to transform local, shallow understanding into holistic, deep comprehension (Kintsch, 1988).

Inspired by this process, we design a streaming thinking paradigm for LLMs, with an example illustrated in Figure 1. At each step, the model incrementally processes the incoming sentence, focusing on progressive comprehension such as (1) understanding and summarizing key information, (2) explaining ambiguities and reorganizing semantic relations, (3) extending logical implications, and (4) skipping thinking step when the content is irrelevant to the question. After completing this incrementally reading and reasoning, we define multiple reasoning depths as a post-reasoning step, as shown in Figure 1 (b). The model may (1) directly produce an answer, representing the shallowest depth; (2) further integrate global information to achieve deeper comprehension; or (3) perform reflective reasoning on top of global integration to obtain the most reliable reasoning outcome. Within the paradigm, reasoning depth adapts to question complexity and is explicitly controlled by instruction signals that guide the model toward different strategies.

Ordering of Context and Question in Streaming Thinking In the batch thinking paradigm, all input is available simultaneously, so the order of context and question is often overlooked. Yet prior work shows that their relative order can influence reasoning (Chen et al., 2024b; Wei et al., 2024; Xie, 2024), an effect amplified in streaming scenarios. In human reading, two natural input orders commonly occur. In the first, the question is presented before the context, enabling the reader to establish targeted associations as subsequent information is processed. In the second, the context precedes the question, in which case the question remains unavailable during streaming reasoning and the reader must rely solely on the context itself to construct plausible inferences. To approximate these scenarios, our streaming thinking paradigm explicitly distinguishes between the two orders and examines their impact on model reasoning. Some examples are provided in Appendix B.

Formal Definition Streaming thinking is defined as an immediate reasoning process that unfolds alongside the input stream, with reasoning depth flexibly adapting to the complexity of the problem. Formally, let Q denote the input question and C_t the t-th sentence in the input context. At each step, the LLM generates an intermediate reasoning state R_t corresponding to C_t, and R_q corresponding to the question Q. The instruction I sets the reasoning depth, directing how intermediate states are integrated into the final reasoning output R. The streaming thinking process can be described as:

$$\mathcal{P}_{\text{streaming}} = \begin{cases} \prod_{t=1}^{T} P(R_t|C_{\le t}, R_{\le t-1}) \cdot P(R_q|Q, C_{\le T}, R_{\le T}) \cdot P(R|Q, C_{\le T}, R_{\le T}, I), & \text{context first,} \\ \underbrace{P(R_q|Q) \cdot \prod_{t=1}^{T} P(R_t|Q, C_{\le t}, R_{\le t-1})}_{\text{streaming thinking}} \cdot \underbrace{P(R|Q, C_{\le T}, R_{\le T}, I)}_{\text{with controllable depth}}, & \text{question first.} \end{cases} \tag{1}$$

3 StreamingThinker

This section introduces the StreamingThinker, a supervised fine-tuning framework that integrates streaming CoT generation with streaming training and inference mechanisms to adapt batch-oriented LLMs to the streaming thinking paradigm.

3.1 Streaming CoT Generation

StreamingThinker first constructs a streaming-like CoT dataset, as existing batch-style reasoning traces lack human-like incremental thinking. This step produces streaming-compatible traces with controllable depth, providing the foundation for subsequent training and evaluation.

Generation Process The streaming reasoning dataset is constructed through a multi-stage pipeline, as shown in Figure 2. We first insert sentence-level boundary tokens `<EOS>` for the input to define minimal reasoning units. Then the LLM is prompted to generate order-preserving reasoning for the preceding sentence and terminate the step with `<EOT>`, when encountering `<EOS>`.

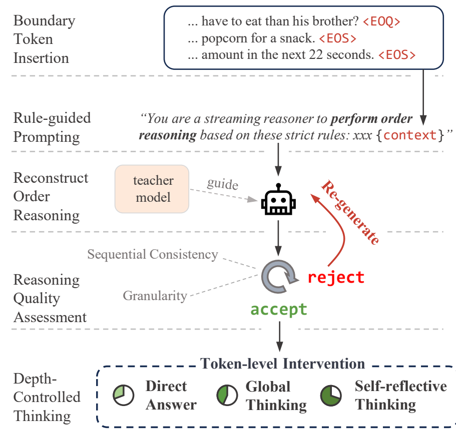

Figure 2: Generation for streaming CoT.

[2] To further enforce sequential alignment, a higher-parameter teacher model reconstructs the generated reasoning. Once all sentence-level reasoning traces are generated, they are evaluated using granularity score and sequential consistency score. Passing samples are enhanced with token-level intervention to generate depth-controlled reasoning variants. Samples that fail the evaluation are regenerated, and those still failing under Pass@2 (Chen et al., 2021) metric are discarded.

Quality Assurance and Evaluation We propose two evaluation metrics: the *granularity score* measures the fine-grained alignment of the streaming CoT, while the *sequential consistency score* assesses whether the reasoning proceeds in a streaming, order-preserving way. The granularity score is defined as the ratio between the number of boundary tokens in the input and those in the output: $granularity = \frac{N_{\text{EOS}}}{N_{\text{EOT}}}$, where N_{EOS} and N_{EOT} denote the counts of boundary tokens in the input and output, respectively. When the granularity score is equal to 1, the reasoning matches the input in boundary count, suggesting ideal alignment. The consistency score is defined as the similarity between the input sentence and reasoning sentences, i.e., $consistency = \text{sim}(R_t, C_t) = \frac{v_R \cdot v_C}{\|v_R\| \, \|v_C\|}$, where v_R and v_C denote the embedding vectors of the reasoning sentences R_t and the input sentence C_t, respectively. We use SentenceBERT (Reimers & Gurevych, 2019) for sequential consistency calculation.[3]

3.2 Streaming Training Framework

A naive approach is to interleave input and reasoning sentences. While it appears to be streaming, this method is inconsistent with the pretraining format of LLMs and still enforces serial execution, where reasoning prevents the model from simultaneously consuming new inputs. Beyond interleaving, we design an actual streaming training framework for streaming thinking paradigm.

Streaming Attention Mask Matrix According to Equation 1, the core constraint of streaming thinking is that the reasoning step at current time must not access future inputs. In contrast, the standard attention mask used for batch thinking exposes all inputs to every reasoning step. To adapt LLMs to the streaming paradigm, we inject a streaming constraint into the attention mask. As illustrated in Figure 3 (a), within the attention from the reasoning sentences to the input sentences, we apply a causal mask that blocks attention from step t to input positions $> t$. We refer to the masked region as the streaming mask region. Let the input sentence have length T and the reasoning segment length L. The streaming mask is then defined as

$$\mathcal{M}_{\text{streaming}}(i,j) = \mathcal{M}(i,j) + \big(-\infty - \mathcal{M}(i,j)\big) \cdot \mathbb{I}_{\{\, i>T,\ j<T,\ j>i-T+1 \,\}}, \qquad (2)$$

where $\mathcal{M}$ is the vanilla casual mask matrix of LLMs, and $\mathbb{I}$ is an indicator function.

Streaming Position Encoding The RoPE (Su et al., 2024) of LLMs represents relative positions by rotating queries and keys, with the attention between a reasoning token R_t and an input token S_t expressed as $Attn(R_t, S_t) = q_R^T R(T + t - t) k_S$, where T is the input length and $t, t+T$ are their positional IDs, and $R(T)$ is the rotary matrix. However, in streaming scenarios, the concurrent generation of output and reception of input induces positional contention in the encoding space (Tong et al., 2025; Guo et al., 2024). To address this issue, we assign independent position IDs to input and reasoning tokens, both starting from zero. Formally, the positional IDs of reasoning token R_t and input token S_t are both set to t, yielding $Attn(R_t, S_t) = q_R^T R(t - t) k_S$. This design removes positional contention in streaming parallel processing. Furthermore, identical position IDs ensure that, during streaming reasoning, a reasoning sentence is positioned nearest to its associated input sentence and distant from others, which conforms to the essential principle of streaming alignment.

[2]Boundary tokens mark minimal reasoning units and indicate the end of a reasoning step during inference.

[3]In cases where a single input sentence corresponds to multiple reasoning sentences, we regard them collectively as one segment. We provide the similarity map between the input and the reasoning in the Appendix C.5.

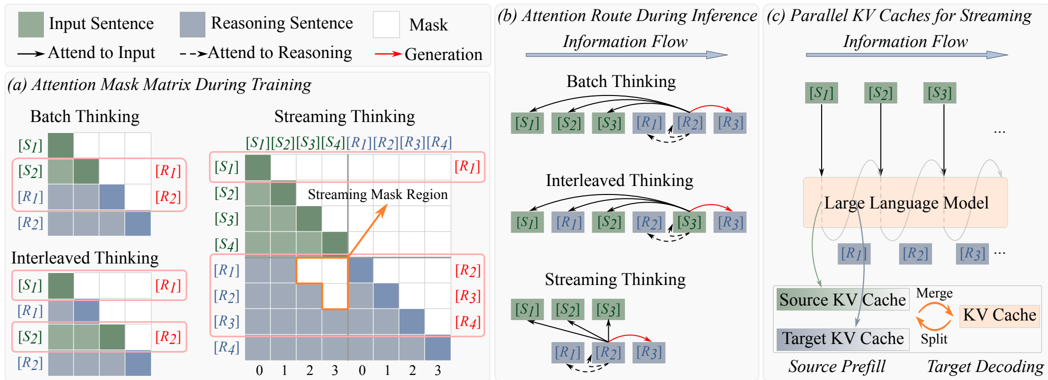

Figure 3: Training and inference framework of StreamingThinker. (a) shows attention mask at training. (b) and (c) show attention routing and parallel KV caches for streaming thinking at inference.

3.3 Streaming Inference

Attention Route Figure 3(b) compares information flow across paradigms. In batch thinking, reasoning begins only after the full context is received, resulting in long attention routes and serial dependency. Interleaved thinking alternates reasoning with partial inputs but still updates a single cache sequentially, and its format diverges from the pretraining distribution. In contrast, streaming attention preserves consistency with batch-style pretraining while employing parallel caches, enabling concurrent processing with shorter routes and lower latency.

Parallel KV caches To enable parallel processing in streaming reasoning, we design two KV caches during inference: a source cache for input tokens and a target cache for reasoning tokens, as shown in Figure 3 (c). As input arrives sentence by sentence, the LLM performs prefill in arrival order, storing hidden states in the source cache. Before decoding, the two caches are merged so that reasoning can attend to the inputs, and newly generated tokens are written into the merged cache. After finishing a sentence, the caches are split again. This design enables concurrency between source-side prefill and target-side decoding, whereas batch and interleaved paradigms rely on a single continuous cache, enforcing strictly serial execution.

4 Experiments

4.1 Experimental Settings

Datasets To evaluate StreamingThinker, we conduct a comprehensive assessment across three representative reasoning tasks: math reasoning, logical reasoning, and context-based QA reasoning. For math reasoning, we selected GSM-symbolic (Mirzadeh et al., 2025) and MetamathQA (Yu et al., 2024). For logical reasoning, we utilized LogicNLI (Tian et al., 2021) and ProofWriter (Tafjord et al., 2020). Finally, for context-based QA reasoning, PubMedQA (Jin et al., 2019) and HotpotQA (Yang et al., 2018) were employed. All datasets were partitioned into dedicated training and testing sets, with detailed specifications provided in the Appendix D.

Models and Baselines We implement our StreamingThinker using models from the Qwen3 family. For streaming CoT generation, we utilize Qwen3-32B as the initial generation model to produce the preliminary streaming reasoning trace. We assign Qwen3-235B-A22B-Instruct as the teacher guidance model to reconstruct the preliminary trace. Then Qwen3-1.7B and Qwen3-4B as the backbone of StreamingThinker for evaluation. To provide a comprehensive evaluation, we compare StreamingThinker with three baselines representing alternative reasoning paradigms: (1) *batch thinking (Batch, orignal)*, where the model generates reasoning after observing the entire context without additional supervision; (2) *batch thinking with CoT distillation (Batch, SFT)*, where reasoning traces are distilled from a stronger 32B teacher model to enhance reasoning ability; and (3) *interleaved mode*, a naive streaming variant that alternates between input segments and reasoning steps without parallel cache support.

Table 1: Pass@1 accuracy (Acc↑) and token usage (Tokens↓) results of the streaming thinking paradigm under the batch processing setting. The comparison includes: (1) the original batch thinking baseline, (2) SFT models trained with RoPE or Streaming RoPE (SPE) distilled from Qwen3-32B CoT data, and (3) the streaming thinking model executed in a batch processing mode (Batch-S) with RoPE or SPE. Reasoning depth is categorized into three levels, denoted as D1–D3, where D1 = direct answer, D2 = with global reasoning, and D3 = with self-reflection.

Methods	Math reasoning				Logical reasoning				Context-based QA reasoning			
	GSM-Symbolic		MetaMathQA		ProofWriter		LogicNLI		HotPotQA		PubMedQA	
	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓
Qwen3-1.7B												
Batch, Original	0.645	1714.25	0.657	1751.40	0.488	1379.58	0.503	1384.47	0.484	638.91	0.616	619.92
Batch, SFT, RoPE	**0.748**	1104.05	0.755	1236.76	0.785	980.12	0.600	1120.38	0.530	512.42	0.642	**602.38**
Batch, SFT, SPE	0.740	1032.71	**0.761**	1195.68	0.790	970.44	**0.604**	**1110.27**	0.528	**498.73**	0.648	611.25
Batch-S, D1, RoPE	0.396	217.33	0.1795	253.26	0.470	360.18	0.515	872.33	0.456	324.71	0.598	452.15
Batch-S, D1, SPE	0.401	212.42	0.183	250.26	0.476	352.07	0.519	865.27	0.463	318.64	0.601	448.92
Batch-S, D2, RoPE	0.688	396.34	0.703	611.61	0.540	555.21	0.542	1196.33	0.492	472.51	0.624	571.66
Batch-S, D2, SPE	0.692	391.25	0.706	618.53	0.544	545.37	0.546	1184.47	0.497	465.33	0.628	565.92
Batch-S, D3, RoPE	0.732	562.88	0.736	**829.58**	0.804	720.46	0.560	1351.16	0.545	589.42	**0.668**	693.18
Batch-S, D3, SPE	0.727	**552.13**	0.738	835.11	**0.810**	**710.32**	0.568	1362.82	**0.552**	583.61	0.663	688.74
Qwen3-4B												
Batch, Original	0.855	1445.61	0.774	1630.67	0.620	1878.57	0.492	1421.92	0.575	617.40	0.609	463.37
Batch, SFT, RoPE	**0.890**	896.45	0.833	1029.66	**0.863**	935.36	0.647	1002.37	0.608	**455.56**	0.675	572.63
Batch, SFT, SPE	0.882	887.14	**0.836**	989.23	0.861	925.66	0.650	**992.38**	0.601	467.21	**0.680**	582.31
Batch-S, D1, RoPE	0.433	203.23	0.662	201.60	0.592	386.25	0.627	683.17	0.552	269.45	0.598	358.14
Batch-S, D1, SPE	0.437	199.64	0.668	200.73	0.596	376.13	0.623	672.61	0.561	257.11	0.592	351.47
Batch-S, D2, RoPE	0.864	348.52	0.791	534.74	0.801	583.28	0.639	983.52	0.592	381.36	0.657	507.32
Batch-S, D2, SPE	0.871	352.17	0.806	528.33	0.795	596.61	0.642	978.52	0.601	388.93	0.651	498.17
Batch-S, D3, RoPE	0.867	499.83	0.821	**661.96**	0.856	721.89	0.645	1242.82	0.616	474.56	0.661	**563.41**
Batch-S, D3, SPE	0.874	**493.26**	0.825	668.19	0.861	**715.26**	**0.651**	1176.65	**0.621**	466.14	0.669	571.22

Metric The evaluation of streaming scenarios can be viewed as a trade-off between performance and latency. For reasoning performance, we adopt Pass@1 score as the accuracy metric to measure the model's ability to successfully solve problems. Latency is assessed at two levels: token latency and time latency. At the token level, we use token-to-first-token (TTFT) to measure how many input tokens must be observed before the model begins reasoning.[4] For time latency, we set the LLM's input speed as the average human speaking rate about 150 words per minute (Geva & Yaghoub Zadeh, 2006; Jacewicz et al., 2010), and define the waiting time until the first answer token as the latency.[5]

4.2 Effectiveness of the Streaming Thinking Paradigm for LLMs

We begin by validating the feasibility of the streaming thinking paradigm (sequential reasoning with depth adjustment) under the batch setting, which removes interference from streaming input and cache strategies. This controlled setup allows us to assess the model's adherence to the paradigm and to further investigate two key aspects: (1) the impact on reasoning trajectories and reasoning depth, and (2) the role of streaming position encoding in maintaining alignment and stability.

Effect of Reasoning Depth in Streaming Thinking We first validate the streaming thinking framework on Qwen3-1.7B and Qwen3-4B. As shown in Table 1 and Figure 4, the performance of LLMs improves consistently with increasing reasoning depth in the streaming paradigm. At shallow depths, the model mainly performs local reasoning aligned with the sequential input, which provides fast but relatively coarse-grained understanding. When deeper reasoning stages are introduced—particularly with global reflection—the performance approaches that of batch thinking, demonstrating that additional depth helps compensate for the information fragmentation inherent in streaming reasoning. Moreover, results in Table 1 show that under the batch processing setting, the

[4]This is similar with time-to-first-token, but measured in terms of token count.

[5]Detailed definitions and additional evaluation metrics are provided in Appendix F.

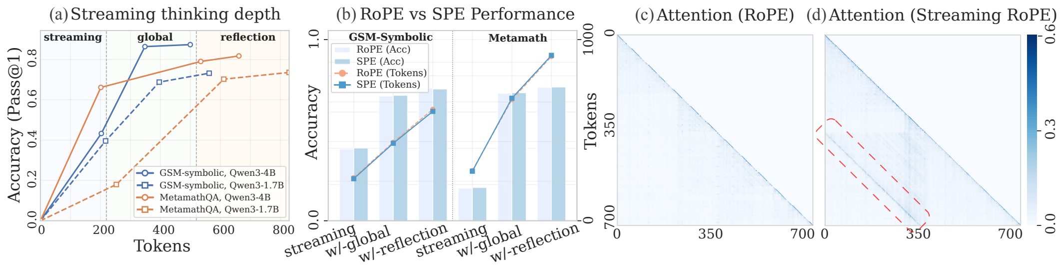

Figure 4: Streaming thinking performance and attention patterns. Subplots (a)–(b) show accuracy–token trade-offs for GSM-Symbolic and MetaMathQA under RoPE and streaming RoPE (SPE); subplots (c)–(d) show average attention maps comparing RoPE and SPE.

streaming thinking paradigm achieves performance comparable to the original batch thinking, while demonstrating notable advantages in token efficiency.

The slope of the curves in Figure 4(a) highlights the marginal gains of each added depth. Notably, introducing the global reasoning stage leads to the most significant improvement, which aligns with the motivation of streaming thinking: since the streaming phase processes inputs in a lightweight, incremental fashion, a global consolidation step is essential to fully integrate dispersed information and support complex inference.

Position Encoding in Streaming Thinking Streaming RoPE assigns consistent yet independent index groups to input and reasoning tokens, ensuring that each reasoning step is correctly aligned with its corresponding input. This design also prevents positional interference, thereby addressing the positional contention issue that arises with the original RoPE in streaming scenarios.

Our experiments further demonstrate that Streaming RoPE achieves performance comparable to the original RoPE under the same settings. As shown in Figure 4(b) and Table 1, in math reasoning tasks at the same depth, Streaming RoPE attains nearly identical accuracy and token consumption to the original RoPE. This indicates that adapting RoPE to the streaming paradigm preserves model capacity without incurring performance degradation.

The attention visualizations in Figures 4(c) and (d) provide additional insights. While the original RoPE exhibits no clear positional preference across the input context, Streaming RoPE shows a pronounced diagonal concentration, reflecting stronger focus on the current context. This bias aligns well with the motivation of streaming thinking: reasoning should primarily rely on information already observed, thereby enabling the model to "*think while reading*."

4.3 LLMs Thinking While Reading Under the Streaming Thinking Paradigm

After confirming the feasibility of streaming thinking in batch settings, we now extend our evaluation to real streaming scenarios, where inputs arrive incrementally over time. In this setting, the model is required to perform reasoning online, relying solely on the partial context available at each step.

Latency of StreamingThinker We examine the latency performance of StreamingThinker using the Qwen3-4B model. As shown in Table 2, streaming thinking achieves a markedly lower TTFT than batch reasoning, as reasoning can be initiated once the first input segment becomes available. The minimal latency observed at depth D1 further indicates that reasoning overlaps with input reading without incurring additional overhead.[6] These results confirm that StreamingThinker substantially reduces response delay, a property of particular importance for streaming applications.

Interleaved mode and Streaming mode The interleaved mode constitutes a naive instantiation of streaming reasoning. Relative to batch reasoning, it exhibits lower latency—most prominently in terms of TTFT—as reasoning can be initiated earlier, as reported in Table 2. Nevertheless, its accuracy is consistently lower and its overall delay higher than those achieved by streaming thinking. This discrepancy can be attributed to the distributional mismatch between interleaved input

[6]It is important to note that the input rate is set to 150 words per minute to match the average speed of human speech in interactive scenarios. Given that LLM decoding operates at a much faster rate, the effective bottleneck in streaming reasoning stems from input arrival, rather than output generation.

Table 2: Results compare the original batch thinking and streaming thinking paradigms. The streaming thinking results include both the naive interleaved streaming mode (Interleaved) and our proposed parallel streaming mode (Streaming), evaluated under three reasoning depths (D1 = direct answer, D2 = global thinking, D3 = self-reflection). Results are reported in terms of Pass@1 (Acc), token number to first input token (TTFT), and time delay for generating first answer token (delay). All experiments are conducted on Qwen3-4B.

Method	GSM-Symbolic			MetaMathQA			ProofWriter		
	Acc↑	TTFT↓	Delay↓ (s)	Acc↑	TTFT↓	Delay↓ (s)	Acc↑	TTFT↓	Delay↓ (s)
Batch, Original	0.855	94.74	47.70	0.774	100.51	53.81	0.620	232.11	61.99
Interleaved, D1	0.410	20.77	6.30	0.655	16.89	6.23	0.583	20.51	11.96
Interleaved, D2	0.829	20.77	11.90	0.754	16.89	16.55	0.750	20.51	18.07
Interleaved, D3	0.843	20.77	15.46	**0.783**	16.89	20.49	0.801	20.51	22.35
Streaming, D1	0.421	20.77	**0.66**	0.657	16.89	**0.68**	0.588	20.51	**0.71**
Streaming, D2	0.842	20.77	5.973	0.752	16.89	10.98	0.761	20.51	6.50
Streaming, D3	**0.856**	**20.77**	**9.768**	0.780	**16.89**	**15.18**	**0.813**	**20.51**	**11.05**
Method	**LogicNLI**			**HotPotQA**			**PubMedQA**		
	Acc↑	TTFT↓	Delay↓ (s)	Acc↑	TTFT↓	Delay↓ (s)	Acc↑	TTFT↓	Delay↓ (s)
Batch, Original	0.492	350.08	46.92	0.575	1485.547	20.37	0.609	357.468	15.29
Interleaved, D1	0.591	42.06	21.17	0.537	24.32	8.33	0.571	21.74	11.09
Interleaved, D2	0.614	42.06	30.47	0.576	24.32	12.02	0.633	21.74	15.71
Interleaved, D3	0.627	42.06	38.50	0.598	24.32	14.45	0.646	21.74	17.45
Streaming, D1	0.603	42.06	**0.72**	0.544	24.32	**0.69**	0.579	21.74	**0.74**
Streaming, D2	0.629	42.06	9.9	0.581	24.32	3.92	0.641	21.74	4.92
Streaming, D3	**0.634**	**42.06**	**18.45**	**0.603**	**24.32**	**6.50**	**0.653**	**21.74**	**6.76**

sequences and the LLMs' pre-training corpus, which impairs reasoning fidelity. Moreover, the interleaved paradigm enforces a sequential synchronization constraint, requiring the completion of ongoing reasoning before additional input tokens can be incorporated, thereby exacerbating latency. In contrast, StreamingThinker employs parallelized KV caches that disentangle input encoding from reasoning generation, enabling concurrent reading and reasoning. This architectural design effectively minimizes latency while preserving reasoning quality, thereby highlighting the necessity of streaming-specific mechanisms for efficient online reasoning.

4.4 Ordering of Context and Question for StreamingThinker

The ordering of context and question plays a critical role in streaming reasoning. Unlike batch settings, where the model has access to the entire input simultaneously, the streaming paradigm requires reasoning to unfold as inputs arrive. In real-world scenarios, however, the order in which the question and context appear is often unknown. To account for this, we evaluate the proposed streaming thinking framework under both orderings to examine its robustness across different streaming conditions.

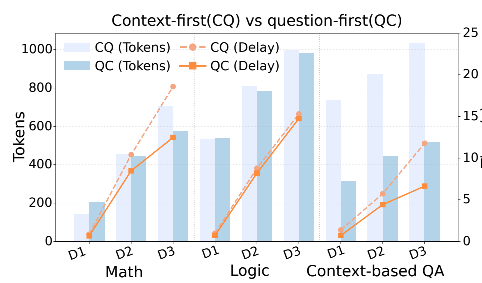

Figure 5: Comparison between context-first and question-first settings. Bars indicate model token consumption, while lines represent time latency.

Table 3 reports the performance of StreamingThinker under the context-first input setting across different datasets. Overall, the results follow a similar trend to the question-first setting, confirming the framework's ability to provide timely responses regardless of input order. As the depth of reasoning increases, both accuracy and latency improve, albeit with a gradual rise in token consumption.

Figure 5 highlights the differences between the two settings.[7] When the question appears first, the model is aware of the reasoning target. This is particularly advantageous in Context-based QA tasks where critical information is sparse; the prior knowledge of the question allows the model to precisely capture key evidence, thereby avoiding the generation of reasoning for irrelevant context (in contrast to domains with denser information). However, for the identified relevant segments, the model tends to expand its logic immediately. This leads to higher token usage at depth D1, and since part of the reasoning is already completed during the streaming process, the incremental

[7]For clarity, we report the average performance for tasks of the same type.

Table 3: Results on context-first setting, where the LLM receives context before the question. (D1 = direct answer, D2 = global thinking, D3 = self-reflection). Results are reported in terms of Pass@1 (Acc), token numbers (Token), token number to first input token (TTFT), and time delay for generating first answer token (delay). All experiments are conducted on Qwen3-4B.

Method	GSM-Symbolic				MetaMathQA				ProofWriter			
	Acc↑	Token↓	TTFT↓	Delay↓	Acc↑	Token↓	TTFT↓	Delay↓	Acc↑	Token↓	TTFT↓	Delay↓
Batch, Original	0.842	1483.65	94.74	48.20	0.774	1668.30	100.51	54.23	0.633	1786.42	232.11	58.05
Streaming, D1	0.476	165.42	18.68	**0.77**	0.668	114.50	20.40	**0.94**	0.528	457.91	8.12	**1.06**
Streaming, D2	0.837	451.81	18.68	9.23	0.760	463.59	20.40	11.59	0.740	697.20	8.12	7.33
Streaming, D3	0.844	797.63	**18.68**	**20.45**	0.788	617.36	**20.40**	**16.71**	0.798	875.19	**8.12**	**13.11**

Method	LogicNLI				HotPotQA				PubMedQA			
	Acc↑	Token↓	TTFT↓	Delay↓	Acc↑	Token↓	TTFT↓	Delay↓	Acc↑	Token↓	TTFT↓	Delay↓
Batch, Original	0.483	1487.56	350.08	48.35	0.566	1492.41	1485.55	48.50	0.593	1320.36	357.47	42.90
Streaming, D1	0.476	604.84	5.60	**0.89**	0.532	1172.36	46.39	**1.62**	0.562	298.51	31.49	**1.12**
Streaming, D2	0.613	927.35	5.60	10.24	0.575	1266.41	46.39	5.90	0.628	475.63	31.49	5.54
Streaming, D3	0.625	1122.32	**5.60**	**16.55**	0.591	1381.73	**46.39**	**12.14**	0.640	687.45	**31.49**	**12.43**

token growth at deeper levels (D2 and D3) becomes smaller compared to the context-first setting. In contrast, when the context appears first, the model lacks knowledge of which information is salient. As a result, its reasoning remains more conservative, proceeding sentence by sentence without extensive elaboration, which produces lower token usage at D1 when processing the same context sentence.[8] However, this characteristic leads to inefficiency in sparse scenarios (such as the context-based QA tasks): due to the conservative strategy, the model fails to skip irrelevant information, resulting in a significantly longer total generation length.

5 Discussion

Efficiency Analysis We evaluate the model efficiency on Qwen3-4B using 100 samples from GSM-Symbolic dataset. As shown in Table 4, the Streaming paradigm reduces the first-token latency (measured by time consumption) from 28.00s to 6.23s ($\sim$4.5$\times$ speedup). Crucially, the parallel KV cache operations introduce negligible temporal overhead, with $split_{kv}$ and $merge_{kv}$ taking less than 5ms combined. Additionally, peak memory usage remains consistent with the baseline ($\sim$7.99 GB). While bandwidth cost increases, this is primarily due to the inherent multiple prefill phases in streaming scenarios rather than the parallel KV cache mechanism itself. Note that Table 4 reports the latency for each stage separately; the actual end-to-end wall-clock time, which benefits from the concurrent execution of reading and reasoning, has been detailed in Section 4.

Why Streaming Thinker Work? Prior studies (Fan et al., 2025; Laban et al., 2025; Li et al., 2025) caution that reasoning over incomplete inputs significantly degrades performance, as standard models are prone to hallucinating based on partial evidence (Fan et al., 2025) or struggling with uncertainty (Laban et al., 2025). However, Streaming Thinker circumvents these pitfalls through three distinct mechanisms. First, global information is **deferred rather than lost**. Unlike scenarios where critical conditions are permanently removed, our framework merely shifts the timing of acquisition, ensuring the model incorporates the full global context after the streaming phase. Second, we employ a **conservative reasoning strategy**. Instead of attempting premature complex reflection, the model concentrates on shallow reasoning" (e.g., intermediate calculations and entity tracking) during the streaming phase. This functions as an incremental pre-processing step that simplifies raw context. Third, this behavior is enforced via a **specialized streaming training paradigm**. Unlike batch models that rigidly apply full-context patterns to partial inputs—often falling into the over-thinking" trap—our model is explicitly adapted to local scopes, learning to process available information without jumping to erroneous conclusions.

[8]The model's conservative reasoning stems from our cautious data generation strategy—when the question is unknown, excessive logical expansion may cause overthinking.

Table 4: Efficiency evaluation on Qwen3-4B. We compare the efficiency of Batch and Streaming paradigms using 100 randomly selected samples from the **GSM-Symbolic** dataset. Metrics include execution memory usage (Peak/Δ), bandwidth cost, and time consumption across different stages.

Metric	Batch Thinking Paradigm			Streaming Thinking Paradigm				
	Read	**Prefill**	**Decoding**	**Read**	**Split KV**	**Prefill**	**Merge KV**	**Decoding**
Count (times)	69.86	1	436.0	69.86	4.65	4.65	4.65	439.0
Peak Mem (MB)	–	7,991	7,998	–	7,957	7,993	7,940	7,999
Total Mem Δ (MB)	–	+222	+102	–	-289	+215	+217	+103
Bandwidth Cost (MB)	–	–	–	–	–	+29,173	+434	-3,310
Avg Time (s)	0.4	0.052	0.039	0.4	< 0.001	0.041	< 0.001	0.037
Total Time (s)	27.945	0.052	17.178	27.945	**0.005**	**0.190**	**0.004**	16.392
First-token Latency (s)	28.003	–	–	**6.231**	–	–	–	–

6 Related Work

Efficient Reasoning in LLMs Prior studies on efficient reasoning in LLMs have mainly focused on three directions: token compression (Aggarwal & Welleck, 2025; Xia et al., 2025; Zhang et al., 2025a), structural quantization and pruning (Liu et al., 2025; Srivastava et al., 2025; Zhao et al., 2025), and efficient decoding (Pan et al., 2025; Liao et al., 2025). Token compression methods such as Tokenskip (Xia et al., 2025) condense CoT into fewer tokens to reduce the inference cost. Structural approaches leverage quantization or pruning to compress model parameters for efficient reasoning such as (Srivastava et al., 2025). Efficient decoding has been explored through parallel sampling of generation paths (Pan et al., 2025) or by using smaller models for speculative prediction (Liao et al., 2025) to reduce inference latency. In contrast, our work introduces streaming processing as a complementary dimension of efficiency, allowing reasoning to proceed concurrently with input processing to reduce response latency.

Streaming LLMs Recent research on streaming LLMs has focused on three directions: architecture adaptation (Tong et al., 2025; Raffel et al., 2024; Guo et al., 2024), latency control (Ma et al., 2018; Ahmed et al., 2025; Cheng et al., 2025), and modality adaptation (Chen et al., 2024a; 2025a; Xie & Wu, 2024). Architecture adaptation addresses the mismatch between decoder-only transformers and streaming settings (Tong et al., 2025), where issues such as positional interference and redundant re-encoding are mitigated through recurrent states, and modified attention mechanisms. Latency control emphasizes fine-grained alignment between input and output, ranging from fixed policies (e.g., wait-k (Ma et al., 2018)) to adaptive scheduling that optimize the accuracy–latency balance. Modality adaptation extends streaming LLMs beyond text to speech recognition (Guo et al., 2024), speech translation (Cheng et al., 2025), and video understanding (Chen et al., 2024a) by integrating modality-specific encoders and synchronization mechanisms. From a reasoning perspective, recent studies (Xie et al., 2025a; Chiang et al., 2025; Xie et al., 2025b) have explored alternating reasoning and generation to approximate streaming operation. Our work differs by explicitly modeling thinking while reading, enabling reasoning to evolve concurrently with incremental input.

7 Conclusion

In this work, we introduce the streaming thinking paradigm for large language models, inspired by the human ability to think while reading. Unlike conventional batch reasoning, this paradigm unfolds reasoning concurrently with input arrival and adapts its depth after reading is complete. To instantiate this paradigm, we developed StreamingThinker, which integrates a streaming CoT generation pipeline, streaming-constrained training, and parallel inference supported by specialized KV cache designs. Comprehensive experiments across math reasoning, logical reasoning, and long-context QA demonstrate that StreamingThinker substantially reduces latency while preserving or improving reasoning quality. Our analysis further reveals the benefits of controllable reasoning depth, streaming-specific position encoding, and parallel inference for enabling true concurrency. These findings highlight streaming thinking as a promising new direction for efficient and coherent reasoning in LLMs, bridging the gap between artificial and human-like cognition.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our work. All experimental settings, including dataset descriptions, training details, and hyperparameter selections, are clearly documented in the main text and appendix. We further provide extensive ablation studies to justify our design choices. Upon acceptance, we will release the full codebase and scripts to facilitate replication.

REFERENCES

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

Zeeshan Ahmed, Frank Seide, Zhe Liu, Rastislav Rabatin, Jachym Kolar, Niko Moritz, Ruiming Xie, Simone Merello, and Christian Fuegen. Non-monotonic attention-based read/write policy learning for simultaneous translation. *arXiv preprint arXiv:2503.22051*, 2025.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18407–18418, 2024a.

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video llm with streaming speech transcription at scale. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29083–29095, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025b.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. *International Conference on Machine Learning (ICML 2024)*, pp. 6596–6620, 2024b.

Shanbo Cheng, Yu Bao, Zhichao Huang, Yu Lu, Ningxin Peng, Lu Xu, Runsheng Yu, Rong Cao, Yujiao Du, Ting Han, et al. Seed liveinterpret 2.0: End-to-end simultaneous speech-to-speech translation with your voice. *arXiv preprint arXiv:2507.17527*, 2025.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. Stitch: Simultaneous thinking and talking with chunked reasoning for spoken language models. *arXiv preprint arXiv:2507.15375*, 2025.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*, 2025.

Esther Geva and Zohreh Yaghoub Zadeh. Reading efficiency in native english-speaking and english-as-a-second-language children: The role of oral proficiency and underlying cognitive-linguistic processes. *Scientific Studies of Reading*, 10(1):31–57, 2006.

Arthur C Graesser, Murray Singer, and Tom Trabasso. Constructing inferences during narrative text comprehension. *Psychological review*, 101(3):371, 1994.

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

Shoutao Guo, Shaolei Zhang, and Yang Feng. Decoder-only streaming transformer for simultaneous translation. *arXiv preprint arXiv:2406.03878*, 2024.

Ewa Jacewicz, Robert Allen Fox, and Lai Wei. Between-speaker and within-speaker variation in speech tempo of american english. *The Journal of the Acoustical Society of America*, 128(2): 839–850, 2010.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

Walter Kintsch. The role of knowledge in discourse comprehension: a construction-integration model. *Psychological review*, 95(2):163, 1988.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, pp. 15339–15353, 2024.

Jinzhe Li, Gengxu Li, Yi Chang, and Yuan Wu. Don't take the premise for granted: Evaluating the premise critique ability of large language models. *arXiv preprint arXiv:2505.23715*, 2025.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems (NeurIPS 2023)*, 36:36407–36433, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL 2024)*, 11:157–173, 2024.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. Quantization hurts reasoning? an empirical study on quantized reasoning models. *arXiv preprint arXiv:2504.04823*, 2025.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, et al. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. *arXiv preprint arXiv:1810.08398*, 2018.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS 2023)*, 36: 46534–46594, 2023.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*, 2025.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *arXiv preprint arXiv:2407.00079*, 2024.

Matthew Raffel, Victor Agostinelli, and Lizhong Chen. Simultaneous masking, not prompting optimization: A paradigm shift in fine-tuning llms for simultaneous translation. *arXiv preprint arXiv:2405.10443*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. Towards reasoning ability of small language models. *arXiv preprint arXiv:2502.11569*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the first-order logical reasoning ability through logicnli. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*, 2021.

Junlong Tong, Jinlan Fu, Zixuan Lin, Yingqi Fan, Anhao Zhao, Hui Su, and Xiaoyu Shen. Llm as effective streaming processor: Bridging streaming-batch mismatches with group position encoding. *arXiv preprint arXiv:2505.16983*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023.

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

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. Unveiling selection biases: Exploring order and token sensitivity in large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 5598–5621, 2024.

Tong Wu, Chong Xiang, Jiachen T Wang, G Edward Suh, and Prateek Mittal. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*, 2025.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025a.

Zhifei Xie and Changqiao Wu. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*, 2024.

Zhifei Xie, Ziyang Ma, Zihang Liu, Kaiyu Pang, Hongyu Li, Jialin Zhang, Yue Liao, Deheng Ye, Chunyan Miao, and Shuicheng Yan. Mini-omni-reasoner: Token-level thinking-in-speaking in large speech models. *arXiv preprint arXiv:2508.15827*, 2025b.

Zikai Xie. Order matters in hallucination: Reasoning order as benchmark and reflexive prompting for large-language-models. *arXiv preprint arXiv:2408.05093*, 2024.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025a.

Yuwei Zhang, Jayanth Srinivasa, Gaowen Liu, and Jingbo Shang. Attention reveals more than tokens: Training-free long-context reasoning with attention-guided retrieval. *arXiv preprint arXiv:2503.09819*, 2025b.

Anhao Zhao, Fanghua Ye, Yingqi Fan, Junlong Tong, Zhiwei Fei, Hui Su, and Xiaoyu Shen. Skipgpt: Dynamic layer pruning reinvented with token awareness and module decoupling. *arXiv preprint arXiv:2506.04179*, 2025.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, 2024.

A Motivation and Prospective Applications

A.1 Motivation and Prospective Applications

One motivation for introducing *streaming thinking* arises from the limitations of the conventional batch reasoning paradigm. In batch reasoning, a model must wait until the entire input is observed before producing any reasoning, which introduces latency, prevents timely responses, and undermines robustness in handling dynamic or sequential information. In contrast, streaming thinking enables large language models to reason incrementally as input arrives, closely mirroring the human cognitive process of *thinking while reading*. A second motivation lies in its broad practical value: this capability unlocks a wide range of applications where timely reasoning and continuous adaptation are essential. We provide some potential applications as follows.

Real-time Dialogue and Interactive Systems In advanced conversational agents or AI tutors, streaming thinking allows the model to perform continuous reasoning on a user's partial utterances. For example, it can infer a user's evolving intent, reason about the logical consistency of their arguments in real-time, and formulate clarifying questions or guidance without waiting for the user to pause. This enables a fluid, collaborative dialogue rather than a static, turn-based exchange.

Long-Context Analysis and Synthesis When processing lengthy documents, live transcripts, or codebases, the model can engage in incremental synthesis. It continuously builds and refines a mental model of the content, reasoning about causal links, logical dependencies, and thematic connections across the information stream. This is crucial for tasks like real-time meeting summarization, where key decisions and action items must be identified and reasoned about as they emerge.

Human-AI Collaborative Reasoning In creative and analytical workflows, streaming thinking facilitates a true partnership. An AI can act as a thought partner, reasoning alongside a human analyst or writer. As the human proposes an idea, the AI can immediately reason about its implications, offer counter-arguments, or synthesize it with prior information, creating a dynamic and interactive brainstorming loop that accelerates discovery and innovation.

Dynamic Decision-Making and Planning In high-stakes environments such as autonomous navigation or real-time financial market analysis, streaming thinking is critical. An autonomous agent must reason about a constantly changing environment from a stream of sensory data to make timely decisions. This involves continuously updating its world model, predicting future states, and re-evaluating its action plans based on the most current information, a process impossible under the latency of a batch thinking paradigm.

Embodied Intelligence and Robotic Control The streaming thinking paradigm is also particularly well-suited for the challenges presented by embodied AI. An embodied agent, like a robot, operates within a dynamic physical environment, which distinguishes it from models that process static information. It continually receives multi-modal sensory data and is required to respond to this information in a timely manner. In this context, streaming thinking allows the agent to engage in continuous perceptual reasoning, interpreting incoming data to consistently update its internal model of the world. This capability supports dynamic instrumental reasoning, where the model can flexibly plan and re-plan its actions to navigate changing conditions and work towards a goal. For example, a household robot would need to reason about multiple factors at once, such as the movement of a person, the delicacy of an object it plans to handle, and the best path through a cluttered space, while adapting its motor commands accordingly. This close coupling of perception, reasoning, and action in a real-time loop is a core aspect of embodied cognition, and achieving it presents significant challenges for a conventional batch thinking approach.

Streaming Multimodal Understanding Beyond text, streaming thinking is vital for interpreting continuous non-verbal data streams, such as live video feeds or audio environments. For instance, in video understanding, a model must reason about the temporal causality of events as they occur—identifying that an action in frame t is a consequence of an event in frame $t - n$. This is essential for applications like live sports commentary, real-time video surveillance for anomaly detection, or accessibility tools that provide live descriptions of the visual world for the visually impaired. By maintaining an evolving memory of the visual stream, the model can provide coherent, context-rich interpretations without the need to process the entire video offline.

A.2 Research Scope and Positioning

This work positions itself as a pioneering exploration into the **paradigm of Streaming Thinking**—technically enabling Large Language Models (LLMs) to perform concurrent reading and reasoning. Our primary contribution lies in establishing the architectural and training methodologies required to adapt LLMs for efficient, continuous data processing.

Regarding our evaluation scope, we utilize mathematical reasoning, logic reasoning, and contextual QA tasks primarily as a controlled testbed to verify the logical consistency of our streaming framework. These deterministic domains serve as a rigorous testbed to verify that our model can maintain logical consistency and accuracy under the strict constraints of streaming inference. Our future work aims to extend this paradigm to complex streaming scenarios.

B Streaming Thinking Paradigm

B.1 Cognitive Foundations of Streaming Thinking

Human reasoning during reading naturally unfolds in a streaming manner—people engage in comprehension and inference as information arrives, without waiting for the complete context.

Our proposed StreamingThinker architecture draws structural inspiration from established models of human discourse comprehension, specifically the Construction-Integration (CI) model proposed by Kintsch (Kintsch, 1988). The CI model posits that comprehension occurs in two alternating cycles:

- The Construction Phase: A bottom-up process where linguistic input triggers the activation of concepts and propositions based on local context. In this stage, the cognitive system acts as a high-bandwidth information buffer, prioritizing the establishment of local coherence (e.g., resolving immediate pronouns or connecting adjacent clauses) over global consistency. Crucially, this phase is non-selective: it allows multiple, potentially conflicting interpretations to co-exist in a temporary cognitive state. This ensures that no critical information is prematurely discarded before the full context is available
- The Integration Phase: Once the initial propositional network is constructed, the cognitive system iteratively updates node activations based on their connectivity and connection strengths. Through this process, the system resolves local ambiguities (e.g., polysemy) and filters out contextually inappropriate inferences. The result is a refined macrostructure where only the information strictly consistent with the global context remains active, ensuring a unified and logical understanding of the discourse.

Inspired by this cognitive process, we proposed in the main text a Streaming Thinking Paradigm that enables large language models to reason concurrently with incremental input. Our streaming phase corresponds to the construction phase, where the model performs ”shallow reasoning” (e.g., entity tracking, intermediate calculation) to process incoming chunks and maintain local coherence (Graesser et al., 1994). Our final reasoning phase corresponds to the integration phase, where the model utilizes the fully accumulated context (KV cache) to synthesize a globally consistent answer. This theoretical alignment explains why our model avoids the ”hallucination” pitfalls of premature guessing: like a human reader, it defers the final integration of complex causal chains until the necessary global information is available.

In this appendix, we further elaborate on this paradigm by providing detailed explanations and illustrative examples that clarify its operational design and the ordering of context and question.

At each step, the model incrementally processes the incoming sentence, focusing on shallow and progressive comprehension, which aligns the construction phase in human comprehension. Specifically, we designed distinct categories of tasks for the model during local construction, serving as cognitive scaffolds for establishing local coherence. These tasks (e.g., intermediate calculation, entity tracking) compel the model to explicitly process and encode immediate details, transforming raw input into structured representations without prematurely committing to a global conclusion. (1) understanding and summarizing key information, (2) explaining ambiguities and reorganizing semantic relations, (3) extending logical implications, and (4) skipping thinking step when the content is irrelevant to the question.

After completing incrementally reading and reasoning, the model shifts its focus from local coherence to global interpretation. It leverages the full context now available in its memory (KV cache) to perform the deep, unconstrained reasoning that was deliberately deferred during the streaming phase. Specifically, the model may: (1) directly produce an answer, representing the shallowest depth; (2) further integrate global information to achieve deeper comprehension; or (3) perform reflective reasoning on top of global integration to obtain the most reliable reasoning outcome.

B.2 Examples of Paradigm Design

We instantiate Streaming Thinking on three representative tasks—math reasoning, logical reasoning, and long-context QA—to show how the paradigm incrementally processes input, manages intermediate reasoning, and enables depth-wise answers (see Box B.1a–c).

Math reasoning (Box B.1a). Given a multi-conditions problem, the model reads the context sentence by sentence, identifying rate/quantity information, skipping irrelevant text, and maintaining interpretable intermediate results (e.g., partial counts). The pipeline supports three reasoning depths: D1 a direct computation from the current notes, D2 a globally consolidated solution that aggregates all intervals, and D3 a reflective pass that re-checks each arithmetic step to ensure reliability.

Paradigm design for Streaming Thinking in Math Reasoning

Input sentences

Question

1. How many more popcorns does Mohammed eat than his brother?

Context

2. Mohammed is popping popcorn for a snack.

3. As the skillet of kernels heats up, the kernels start popping faster.

4. 100 pop in the first 22 seconds of cooking, then 3 times that amount in the next 22 seconds.

5. The kernels increase to 6 times the initial popping rate in the next 22 seconds.

6. In the final 22 seconds, the popping slows down to half the rate as the past 22 seconds.

7. After Mohammed takes the skillet off the heat, a quarter of the number of kernels that popped in the final 22 seconds of cooking also pop from the residual heat.

8. Mohammed gives 2/5 of the made popcorn to his brother Yusuf.

Streaming thinking

<think>

1. Okay, I need to calculate how many more pieces of popcorn Mohammed has than his brother.

2. <Skip>

3. Heating the skillet accelerates kernel popping.

4. The process starts with 100 kernels popping in the first 22-second interval, followed by a second interval where the amount triples, yielding 100×3=300 kernels.

5. In the next 22 seconds, the rate increases to 6 times the initial rate, meaning the number of popped kernels is based on the initial amount of 100, resulting in 100×6=600 kernels for this third period.

6. For the final 22 seconds of cooking, the rate slows to half of the preceding interval's rate, which gives us 600÷2=300 kernels.

7. After the skillet is removed from the heat, residual heat pops an additional quarter of the amount from the final cooking period, adding 300÷4=75 kernels.

8. Finally, the total amount of popcorn is the sum of all intervals: 100+300+600+300+75=1375. Mohammed gives 2/5 of this to his brother Yusuf, which amounts to 1375×2÷5=550 kernels.

[***Optional depth, streaming thinking with direct answer***]

So, the answer is 275.

[***Optional depth, streaming thinking with global thinking***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275.

[***Optional depth, streaming thinking with global thinking & reflection***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275. **Wait, let me check again.** The first interval is 100, second is 3× that = 300, third is 6× the initial rate (not cumulative), so 600—that's right. Fourth is half of the previous interval's rate, so half of 600 is 300, yes. Then residual heat adds a quarter of that final 300, so 75—correct. Total: 100+300+600+300+75=1,375. He gives 2/5 to Yusuf: 1,375×2÷5=550. That leaves Mohammed with 825. Difference: 825-550=275. All steps check out. So yes, Mohammed has 275 more pieces than his brother.

<\think>

Logical reasoning (Box B.1b). Premises are normalized as explicit statements; each incoming sentence is mapped to entities and relations, while unrelated sentences are skipped. Local implications are updated online, and the final decision is produced at different depths: D1 yields an immediate verdict when premises suffice; D2 integrates all premises to assess entailment vs. contradiction; D3 performs a reflective consistency check, which may revise a premature “True/False” to “Unknown” when a critical premise (e.g., a location) is not actually supported by the context.

Paradigm design for Streaming Thinking in Logic Reasoning

Input sentences

Question

1. Is the conclusion true given the premises?, [’True’, ’False’, ’Unknown’], conclusion:William Dickinson went to schools located in the United Kingdom for both high school and university.

Context

2.William Dickinson was a British politician who sat in the House of Commons William Dickinson attended Westminster school for high school and then the University of Edinburgh.

3. The University of Edinburgh is a university located in the United Kingdom.

4. William Dickinson supported the Portland Whigs.

5. People who supported the Portland Whigs did not get a seat in the Parliament.

Streaming thinking

`<think>`

1. Okay, the question is asking whether the conclusion — that William Dickinson went to schools in the United Kingdom for both high school and university — is true, based on the given premises. The possible answers are True, False, or Unknown.

2. William Dickinson attended Westminster School for high school and then the University of Edinburgh.

3. The University of Edinburgh is located in the United Kingdom.

4. `<Skip>`

5. `<Skip>`

[***Optional depth, streaming thinking with direct answer***]

Therefore, the answer is True.

[***Optional depth, streaming thinking with global thinking***]

The conclusion specifies that both his high school and university were located in the United Kingdom. The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. Therefore, the answer is Unknown.

[***Optional depth, streaming thinking with global thinking & reflection***]

The conclusion specifies that both his high school and university were located in the United Kingdom. The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. **Wait, let me check again.** The context only mentions the University of Edinburgh’s location in the UK. Westminster School is not explicitly stated to be in the UK. Even though common knowledge might suggest it is, the premises provided do not confirm this. Therefore, the conclusion cannot be fully verified from the given premises. Therefore, the answer is Unknown.

`<\think>`

Context-based QA (Box B.1c). For scientific or document-level passages, the model streams through long context, filtering sentences that are irrelevant, extracting mechanistic cues (e.g., causal roles, experimental outcomes), and merging compatible evidence across distant spans. D1 gives a concise answer when a single decisive statement appears; D2 synthesizes multi-sentence evidence for a stronger justification; D3 runs a reflective verification that cross-checks all cited evidence for agreement before committing to the final answer.

Paradigm Design for Streaming Thinking in Context-based QA Reasoning

Input sentences

Question

1. Do mitochondria play a role in remodelling lace plant leaves during programmed cell death?

Context

2. Programmed cell death (PCD) is the regulated death of cells within an organism.

3. The lace plant (Aponogeton madagascariensis) produces perforations in its leaves through PCD.

4. The leaves of the plant consist of a latticework of longitudinal and transverse veins enclosing areoles.

5. PCD occurs in the cells at the center of these areoles and progresses outwards, stopping approximately five cells from the vasculature.

6. The role of mitochondria during PCD has been recognized in animals; however, it has been less studied during PCD in plants.

7. The following paper elucidates the role of mitochondrial dynamics during developmentally regulated PCD in vivo in A. madagascariensis. A single areole within a window stage leaf (PCD is occurring) was divided into three areas based on the progression of PCD; cells that will not undergo PCD (NPCD), cells in early stages of PCD (EPCD), and cells in late stages of PCD (LPCD).

8. Window stage leaves were stained with the mitochondrial dye MitoTracker Red CMXRos and examined.

9. Mitochondrial dynamics were delineated into four categories (M1-M4) based on characteristics including distribution, motility, and membrane potential ($\Delta\Psi_m$).

10. A TUNEL assay showed fragmented nDNA in a gradient over these mitochondrial stages.

11. Chloroplasts and transvacuolar strands were also examined using live cell imaging.

12. The possible importance of mitochondrial permeability transition pore (PTP) formation during PCD was indirectly examined via in vivo cyclosporine A (CsA) treatment.

13. This treatment resulted in lace plant leaves with a significantly lower number of perforations compared to controls, and that displayed mitochondrial dynamics similar to that of non-PCD cells.

Streaming thinking

<think>

1. Okay, let's see. The question asks whether mitochondria contribute to the remodeling of lace plant leaves during programmed cell death. I need to focus on 'mitochondria', 'remodeling of lace plant leaves', and 'programmed cell death'.

2. The first sentence introduce that Programmed cell death (PCD) is the regulated death of cells within an organism.

3. In lace plants, leaf perforations are created through programmed cell death.

4. <Skip>

5. <Skip>

6. This study investigates mitochondrial involvement in developmental cell death in lace plant leaves.

7. Researchers divided an areole into three zones: non-PCD cells, early PCD cells, and late PCD cells, reflecting the progression of cell death.

8. They stained mitochondria in window-stage leaves with MitoTracker Red to observe their properties.

9. Mitochondria were classified into four dynamic states (M1–M4) based on distribution, motility, and membrane potential.

10. A DNA fragmentation assay (TUNEL) showed nuclear degradation corresponding with mitochondrial changes, linking mitochondrial dynamics to PCD progression.

11. <Skip>

12. The potential involvement of the mitochondrial permeability transition pore was tested pharmacologically with cyclosporine A.

13. Treatment reduced the number of leaf perforations and preserved mitochondria in a state similar to healthy cells, showing that blocking mitochondrial pore activity suppressed PCD.

[***Optional depth, streaming thinking with direct answer***]

Therefore, the answer is No.

[***Optional depth, streaming thinking with global thinking***]

The study demonstrates that mitochondria change in distribution, motility, and membrane potential during the stages of programmed cell death in lace plant leaves. DNA fragmentation corresponds with these mitochondrial states, and inhibition of the mitochondrial permeability transition pore reduces perforation formation. Together, this evidence shows that mitochondria are actively involved in regulating and executing cell death that sculpts lace plant leaves. Therefore, the answer is Yes.

[***Optional depth, streaming thinking with global thinking & reflection***]

The study demonstrates that mitochondria change in distribution, motility, and membrane potential during the stages of programmed cell death in lace plant leaves. DNA fragmentation corresponds with these mitochondrial states, and inhibition of the mitochondrial permeability transition pore reduces perforation formation. Together, this evidence shows that mitochondria are actively involved in regulating and executing cell death that sculpts lace plant leaves. **Wait, let me check again.** The lace plant forms leaf perforations through PCD, so if mitochondria affect PCD, they affect remodeling. The study showed mitochondria shift in distribution, motility, and membrane potential during PCD stages, changes that align with DNA fragmentation. Blocking the mitochondrial permeability transition pore reduced perforations and preserved healthy-like mitochondria. Thus, mitochondria actively regulate and execute PCD in shaping lace plant leaves. Therefore, the answer is Yes.

<\think>

B.3 Examples of the Ordering of Context and Question

Math reasoning (Box B.2a). *Question-first* (left panel) anchors the target quantity upfront, so each incoming sentence is parsed for rates/amounts relative to the goal, with irrelevant lines skipped and partial counts maintained online. *Context-first* (right panel) encourages incremental accumulation of numerical evidence without a declared target; once the question appears, the model consolidates cached quantities to compute the final difference. Both settings yield interpretable traces but emphasize goal-driven vs. evidence-driven processing, respectively.

Example: Order of Context and Question for Streaming Thinking in Math Reasoning

Question first

Input sentences

Question

1. How many more popcorns does Mohammed eat than his brother?

Context

2. Mohammed is popping popcorn for a snack.

3. As the skillet of kernels heats up, the kernels start popping faster.

4. 100 pop in the first 22 seconds of cooking, then 3 times that amount in the next 22 seconds.

5. The kernels increase to 6 times the initial popping rate in the next 22 s.

6. In the final 22 seconds, the popping slows down to half the rate as the past 22 seconds.

7. After Mohammed takes the skillet off the heat, a quarter of the number of kernels that popped in the final 22 seconds of cooking also pop from the residual heat.

8. Mohammed gives 2/5 of the made popcorn to his brother Yusuf.

Streaming thinking

`<think>`

1. Okay, I need to calculate how many more pieces of popcorn Mohammed has than his brother.

2. `<Skip>`

3. Heating the skillet accelerates kernel popping.

4. The process starts with 100 kernels popping in the first 22-second interval, followed by a second interval where the amount triples, yielding 100×3=300 kernels.

5. In the next 22 seconds, the rate increases to 6 times the initial rate, meaning the number of popped kernels is based on the initial amount of 100, resulting in 100×6=600 kernels for this third period.

6. For the final 22 seconds of cooking, the rate slows to half of the preceding interval's rate, which gives us 600÷2=300 kernels.

7. After the skillet is removed from the heat, residual heat pops an additional quarter of the amount from the final cooking period, adding 300÷4=75 kernels.

8. Finally, the total amount of popcorn is the sum of all intervals: 100+300+600+300+75=1375. Mohammed gives 2/5 of this to his brother Yusuf, which amounts to 1375×2÷5=550 kernels.

[***Optional depth, streaming thinking with direct answer***]

So, the answer is 275.

[***Optional depth, streaming thinking with global thinking***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275.

[***Optional depth, streaming thinking with global thinking & reflection***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275. **Wait, let me check again.** The first interval is 100, second is 3× that = 300, third is 6× the initial rate (not cumulative), so 600—that's right. Fourth is half of the previous interval's rate, so half of 600 is 300, yes. Then residual heat adds a quarter of that final 300, so 75—correct. Total: 100+300+600+300+75=1,375. He gives 2/5 to Yusuf: 1,375×2÷5=550. That leaves Mohammed with 825. Difference: 825-550=275. All steps check out. So yes, Mohammed has 275 more pieces than his brother.

`<\think>`

Context first

Input sentences

Context

1. Mohammed is popping popcorn for a snack.

2. As the skillet of kernels heats up, the kernels start popping faster.

3. 100 pop in the first 22 seconds of cooking, then 3 times that amount in the next 22 seconds.

4. The kernels increase to 6 times the initial popping rate in the next 22 s.

5. In the final 22 seconds, the popping slows down to half the rate as the past 22 seconds.

6. After Mohammed takes the skillet off the heat, a quarter of the number of kernels that popped in the final 22 seconds of cooking also pop from the residual heat.

7. Mohammed gives 2/5 of the made popcorn to his brother Yusuf.

Question

8. How many more popcorns does Mohammed eat than his brother?

Streaming thinking

`<think>`

1. Okay, the input sentences are start with that Mohammed pops popcorn for a snack.

2. Then we know that popping speeds up as the skillet heats.

3. Then the process starts with 100 kernels popping in the first 22-second interval, followed by a second interval where the amount triples, yielding 100×3=300 kernels.

4. In the next 22 seconds, the rate increases to 6 times the initial rate, meaning the number of popped kernels is based on the initial amount of 100, resulting in 100×6=600 kernels for this third period.

5. For the final 22 seconds of cooking, the rate slows to half of the preceding interval's rate, which gives us 600÷2=300 kernels.

6. After the skillet is removed from the heat, residual heat pops an additional quarter of the amount from the final cooking period, adding 300÷4=75 kernels.

7. Finally, the total amount of popcorn is the sum of all intervals: 100+300+600+300+75=1375. Mohammed gives 2/5 of this to his brother Yusuf, which amounts to 1375×2÷5=550 kernels.

8. Now I have read the question and I need to calculate how many more pieces of popcorn Mohammed has than his brother.

[***Optional depth, streaming thinking with direct answer***]

So, the answer is 275.

[***Optional depth, streaming thinking with global thinking***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275.

[***Optional depth, streaming thinking with global thinking & reflection***]

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275. **Wait, let me check again.** The first interval was 100, then three times that → 300, then six times the initial rate (not cumulative) → 600. After that, half of the previous 600 → 300, and finally the residual quarter of that 300 → 75. Adding them all, 100 + 300 + 600 + 300 + 75 = 1375, consistent. Giving 2/5 to Yusuf: 1375 × 2 ÷ 5 = 550. Subtracting leaves 825, and the difference indeed comes out as 275. So yes, after rechecking, the conclusion holds—Mohammed ends up with 275 more pieces than his brother.

`<\think>`

Logical reasoning (Box B.2b). Under *question-first* ordering, the model enters a hypothesis-testing mode: each premise is checked against the claimed conclusion, distractors are skipped, and if a critical premise is missing, depth D3 returns UNKNOWN. With *context-first* ordering, the model first normalizes premises into entities and relations; after the question is posed, it adjudicates entailment/contradiction/unknown using the cached structure. Answers remain consistent across orders, while the traces reveal different timing of inference.

Example: Order of Context and Question for Streaming Thinking in Logical Reasoning

Question first

Input sentences

Question

1. Is the conclusion true given the premises?, ['True', 'False', 'Unknown'], conclusion:William Dickinson went to schools located in the United Kingdom for both high school and university.

Context

2. William Dickinson was a British politician who sat in the House of Commons William Dickinson attended Westminster school for high school and then the University of Edinburgh.

3. The University of Edinburgh is a university located in the UK.

4. William Dickinson supported the Portland Whigs.

5. People who supported the Portland Whigs did not get a seat in the Parliament.

Streaming thinking

<think>

1. Okay, the question is asking whether the conclusion — that William Dickinson went to schools in the United Kingdom for both high school and university — is true, based on the given premises. The possible answers are True, False, or Unknown.

2. William Dickinson attended Westminster School for high school and then the University of Edinburgh.

3. The University of Edinburgh is located in the United Kingdom.

4. <Skip>

5. <Skip>

[***Optional depth, streaming thinking with direct answer***]

So, the answer is 275.

[***Optional depth, streaming thinking with global thinking***]

The conclusion specifies that both his high school and university were located in the UK. The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. Therefore, the answer is Unknown.

[***Optional depth, streaming thinking with global thinking & reflection***]

The conclusion specifies that both his high school and university were located in the United Kingdom. The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. Wait, let me check again. The context only mentions the University of Edinburgh's location in the UK. Westminster School is not explicitly stated to be in the UK. Even though common knowledge might suggest it is, the premises provided do not confirm this. Therefore, the conclusion cannot be fully verified from the given premises. Therefore, the answer is Unknown.

<\think>

Context first

Input sentences

Context

1. William Dickinson was a British politician who sat in the House of Commons William Dickinson attended Westminster school for high school and then the University of Edinburgh.

2. The University of Edinburgh is a university located in the UK.

3. William Dickinson supported the Portland Whigs.

4. People who supported the Portland Whigs did not get a seat in the Parliament.

Question

5. Is the conclusion true given the premises?, ['True', 'False', 'Unknown'], conclusion:William Dickinson went to schools located in the United Kingdom for both high school and university.

Streaming thinking

<think>

1. Okay, we now known that William Dickinson attended Westminster School for high school and then the University of Edinburgh.

2. And, the University of Edinburgh is a university situated in the United Kingdom.

3. William Dickinson supports the Portland Whigs.

4. Then, according to the rule, anyone who supported the Portland Whigs did not obtain a seat in Parliament. So William Dickinson did not obtain a seat in Parliament.

5. Now, the question is asking whether the conclusion — that William Dickinson went to schools in the United Kingdom for both high school and university — is true.

[***Optional depth, streaming thinking with direct answer***]

Therefore, the answer is True.

[***Optional depth, streaming thinking with global thinking***]

The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. Therefore, the answer is Unknown.

[***Optional depth, streaming thinking with global thinking & reflection***]

The context explicitly states he attended Westminster School and the University of Edinburgh. Since the University of Edinburgh is confirmed to be in the UK, the university part is satisfied. However, the location of Westminster School is not directly stated in the context. Westminster School is a well-known school in London, UK, but the context does not explicitly mention its location. Without explicit confirmation that Westminster School is in the UK, the conclusion about both institutions being in the UK remains uncertain. Wait, let me check again. The context only mentions the University of Edinburgh's location in the UK. Westminster School is not explicitly stated to be in the UK. Even though common knowledge might suggest it is, the premises provided do not confirm this. Therefore, the conclusion cannot be fully verified from the given premises. Therefore, the answer is Unknown.

<\think>

C STREAMING COT GENERATION

C.1 DOES INCREMENTAL SENTENCE INPUT WORK?

A straightforward idea is to feed sentences into LLMs sequentially and expect sentence-by-sentence reasoning based on the accumulated input. However, our experiments show that when given only a single sentence without streaming fine-tuning, the original reasoning model tends to overthink (Sui et al., 2025). In other words, although this sentence-level data generation method appears to align better with the streaming property, the vanilla model lacks such generative capability and exhibits reduced controllability in CoT generation.

C.2 DOES DIRECT PROMPTING OF LLMS WORK?

Another intuitive approach is to directly prompt LLMs to perform streaming-style reasoning with carefully crafted instructions. In this setup, the model is explicitly asked to reason sentence by sentence. Nevertheless, our experiments show that, lacking fine-grained control, prompt engineering alone is insufficient: although the model may superficially follow the instructions, it often deviates from the intended trajectory, resulting in inconsistent or overly verbose reasoning. Therefore, we design a pipeline for CoT trajectory control based on the complete input, which enforces sequential reasoning in LLMs through explicit boundary constraints and teacher guidance.

C.3 CONTROL COT PROCESS

In this work, we design a multi-stage pipeline for streaming CoT generation and control. The pipeline consists of four key components: (1) boundary-token insertion to finely define reasoning units and ensure sequential alignment; (2) instruction-based prompting to guide the LLM toward the desired streaming paradigm; (3) teacher-model guidance to reconstruct reasoning chains with improved structural consistency and semantic fidelity; and (4) token-level intervention to control the trajectory of CoT, enabling adjustable reasoning depth and style.

Boundary Token Insert In our proposed streaming thinking paradigm, a complete sentence serves as a atomic unit of cognition. This design achieves a critical balance between the fine-grained nature of streaming processing and the preservation of semantic integrity in the information being processed. To implement this design, we introduce a structured protocol of boundary tokens to explicitly demarcate the model's thinking boundaries. Specifically, this protocol distinguishes between input signals and the model's generated reasoning states:

- Input Demarcation: We employ two distinct tokens to structure the input stream. The `<EOS>` (End-of-Sentence) token follows each contextual sentence, triggering an incremental thinking step for context assimilation. In contrast, the `<EOQ>` (End-of-Question) token marks the end of the problem description, signaling a shift from context processing to answer formulation.
- Thinking State Demarcation: The model's thought process is, in turn, segmented by three corresponding output tokens. The `<EOT>` (End-of-Thought) token concludes each incremental reasoning segment associated with a context sentence. The `<EOQ>` token concludes the reasoning phase triggered by the question. Finally, the `<EOR>` (End-of-Reasoning) token signifies the completion of the entire streaming thinking chain with controllable depth, preceding the final answer generation.

It is important to note that the boundary tokens are introduced primarily as a data generation and evaluation methodology. Their purpose is to provide a structured prompt for the LLMs to generate the desired streaming chain-of-thought behavior. Moreover, in real streaming reasoning scenarios, once the LLM outputs a boundary token, the current reasoning unit can be regarded as completed. Thus, during training, boundary tokens serve as supervisory signals that guide the model to recognize when a reasoning unit should be terminated.

There is an example for boundary tokens insert.

Boundary Tokens Insert for Generation

Input sentences

Question

1. How many more popcorns does Mohammed eat than his brother?<EOQ>

Context

2. Mohammed is popping popcorn for a snack.<EOS>
3. As the skillet of kernels heats up, the kernels start popping faster.<EOS>
4. 100 pop in the first 22 seconds of cooking, then 3 times that amount in the next 22 seconds.<EOS>
5. The kernels increase to 6 times the initial popping rate in the next 22 seconds.<EOS>
6. In the final 22 seconds, the popping slows down to half the rate as the past 22 seconds.<EOS>
7. After Mohammed takes the skillet off the heat, a quarter of the number of kernels that popped in the final 22 seconds of cooking also pop from the residual heat.<EOS>
8. Mohammed gives 2/5 of the made popcorn to his brother Yusuf.<EOS>

Streaming thinking

<think>

1. Okay, I need to calculate how many more pieces of popcorn Mohammed has than his brother.<EOQ>
2. <Skip><EOT>
3. <Skip><EOT>
4. The process starts with 100 kernels popping in the first 22-second interval, followed by a second interval where the amount triples, yielding 100×3=300 kernels.<EOT>
5. In the next 22 seconds, the rate increases to 6 times the initial rate, meaning the number of popped kernels is based on the initial amount of 100, resulting in 100×6=600 kernels for this third period.<EOT>
6. For the final 22 seconds of cooking, the rate slows to half of the preceding interval's rate, which gives us 600÷2=300 kernels.<EOT>
7. After the skillet is removed from the heat, residual heat pops an additional quarter of the amount from the final cooking period, adding 300÷4=75 kernels.<EOT>
8. Finally, the total amount of popcorn is the sum of all intervals: 100+300+600+300+75=1375. Mohammed gives 2/5 of this to his brother Yusuf, which amounts to 1375×2÷5=550 kernels.<EOT>

So, after giving 550 pieces to his brother, Mohammed retains 1375-550=825. The difference is 825-550=275.<EOR>

<\think>

Table 1: Streaming reasoning instruct template.

Input: question, context, example.

You are a streaming reasoner to perform an order reasoning based on the following Question and Context:
Input Question:{`question`}
Input Context:{`context`}

Follow these strict instructions:
1. Must process the Context in order, as if the input arrive in a streaming way.
(1) Must perform reasoning one sentence at a time, i.e. When you encounter an <EOS> in the Input Context, perform reasoning about the sentence that before this <EOS>.
(2) Upon encountering <EOS>, immediately reason about the preceding sentence and end with an <EOT>.
(3) Do Not skip any <EOS> and Do Not forget any <EOT>, i.e. each <EOS> in the Input must match a <EOT> in the reasoning.

2. If you read any Question end with <EOQ>.
(1) Explain the Question that before the <EOQ> in the Input Question. (2) End your interpretation with <EOQ>. (3) Do not add assumptions beyond the explicit meaning of the Question.

3. For each input sentence, smoothly integrate the following into a coherent paragraph-style output:
(1) **Understanding and summarizing key information**: (1-1) Ignore redundant, irrelevant, or ambiguous information. (1-2)Focus on extracting core facts, quantities, conditions, roles, time, location, and other essential elements.
(2) **Explaining ambiguities and reorganizing semantic relation**: (2-1) Clarify vague expressions, ambiguities, or implied information. (2-2) Rephrase the original meaning using clearer, more straightforward, and structurally explicit language, without changing the factual content.
(3) **Lightly extending logical implications**: (3-1) Allowed examples include: Simple calculations (e.g., totals, differences, ratios). Natural inferences about time, sequence, or spatial order. Direct factual inference that logically follows from explicit information. (3-2) Prohibited actions: assumptions, predictions, or subjective evaluations.
(4) **Skipping irrelevant input**: If a sentence is not related to the Question, output <Skip><EOT> immediately after reading <EOS>.

4. **After reading and reasoning all sentencess: DO NOT summarize or infer any judgment.**

There is an in-context example:{`example`}.

Prompt Template The prompt presented in Table 1 casts the model as a streaming reasoner that consumes the context sentence-by-sentence, keyed by `<EOS>`, and emits an immediate reasoning step ending with `<EOT>`. It first interprets any question ending with `<EOQ>` and closes that interpretation with `<EOQ>`. Each step must extract core facts, clarify ambiguities, and permit only light, local inferences; irrelevant sentences are output as `<Skip><EOT>`. No global summary is allowed, and an in-context example anchors the desired behavior.

Teacher Guidance Template The prompt shown in Table 2 directs a teacher LLM to realign an initial streaming trace so that every input sentence (terminated by `<EOS>`) maps exactly to one reasoning step (terminated by `<EOT>`). If present, the question is explained once up to `<EOQ>` before processing the context in strict order. The teacher may only simplify, clarify, and streamline existing content—no new reasoning, no cross-sentence references, no merges, skips, or delays—thereby enforcing strict locality and one-to-one pairing.

Table 2: Instruction template for teacher LLM under both question-first and context-first settings.

Input: question, context, example, reasoning.

You have to reconstruct the following reasoning process based on the Original Input and the Initial Streaming Reasoning Process, ensuring full compliance with the requirements of a streaming reasoner. The Original Input consists of multiple sentences, each ending with <EOS>. Every sentence represents an independent unit of reasoning. Your objective is to revise the initial reasoning such that each reasoning step corresponds exactly.

Follow these strict instructions:

1. If any Question before reading the Input context.

(1) Explain the Question that before the <EOQ> in the Input Question at first.

(2) End your interpretation with <EOQ>.

(3) Do not add assumptions beyond the explicit meaning of the Question.

2. Must process the Context in order, as if the input arrive in a streaming way.

(1) Must perform reasoning one sentence at a time, i.e. When you encounter an <EOS> in the Input Context, perform reasoning about the sentence that before this <EOS>.

(2) Upon encountering <EOS>, immediately reason about the preceding sentence and output a result ending with an <EOT>.

(3) Do Not skip any <EOS> and Do Not forget any <EOT>, i.e. each <EOS> in the Input must match a <EOT> in the reasoning.

3. For each input sentence, smoothly integrate the following into a coherent paragraph-style reasoning output:

(1) Understanding and summarizing key information.

(2) Explaining ambiguities and reorganizing semantic relation.

(3) Lightly extending logical implications.

(4) Skipping irrelevant input.

4. For each sentence:

(1) Do not merge, skip, or delay reasoning for any sentence. Each input sentence must be immediately followed by its reasoning step.

(2) Do not quote or repeat the input sentence in the reasoning output.

(3) Do not introduce new reasoning not found in the original streaming process. You may only simplify, clarify, or streamline what is already there.

(4) Your reasoning must be strictly local to each sentence. Do not refer to previous or later sentences. Avoid speculation, assumptions, or global summaries when streaming reasoning.

There is an in-context example:{example}.

Now, begin your streaming reasoning reconstruction:

Input question: {question}

Input context: {context}

Reasoning content: {reasoning}

Table 3: Instruct template of streaming thinking intervention.

Streaming thinking with direct answer
<|im_start|>user\n {Instruction & Entire Input}<|im_end|>\n   <|im_start|>assistant\n<think>{Streaming Thinking Content}. Now, let me direct output the answer.
Streaming thinking with global thinking
<|im_start|>user\n {Instruction & Entire Input}<|im_end|>\n   <|im_start|>assistant\n<think>{Streaming Thinking Content}. Now, let me start the global thinking, focus on high-level reasoning trace that leads to the final answer.
Streaming thinking with global thinking & reflection
<|im_start|>user\n {Instruction & Entire Input}<|im_end|>\n   <|im_start|>assistant\n<think>{Streaming Thinking Content}Now, let me start the global thinking. {Global Thinking Content}. Wait, let me check again.

Thinking Intervention Template These prompts demonstrated in Table 3 modulate how the assistant transitions from streaming thoughts (<think>) to an answer: (i) direct answer ends the streaming trace and outputs the result immediately; (ii) global thinking triggers a brief high-level consolidation before answering; (iii) global thinking & reflection adds a final self-check pass. All variants preserve the streaming trace while explicitly steering when and how the final answer is produced.

C.4 Quality Evaluation of Streaming CoT

As described in the main text, we evaluate the quality of data generation using two metrics: *granularity score* and *sequential consistency score*. The granularity score ensures that each input sentence corresponds to a distinct and non-overlapping reasoning segment, while the sequential consistency score verifies that the reasoning process unfolds in the same order as the input sequence. In this appendix, we provide a detailed explanation of the two evaluation metrics.

The *granularity score* measures the fine-grained alignment between the segmentation of the input context and that of the reasoning process. Formally, it is defined as the ratio between the number of boundary tokens in the input and those in the output:

$$\text{granularity} = \frac{N_{\text{EOS}}}{N_{\text{EOT}}},$$

where N_{EOS} and N_{EOT} denote the counts of boundary tokens in the input and output, respectively. A score of 1 indicates that the reasoning preserves the segmentation of the input exactly, suggesting ideal boundary alignment.

The *sequential consistency score* evaluates whether reasoning follows the input order in a streaming, segment-by-segment manner. We compute the semantic similarity between each input sentence and its corresponding reasoning segment:

$$\text{consistency} = \text{sim}(R_t, C_t) = \frac{v_R \cdot v_C}{\|v_R\| \, \|v_C\|},$$

where v_R and v_C denote the embedding vectors of reasoning sentence R_t and input sentence C_t, respectively. We employ SentenceBERT (Reimers & Gurevych, 2019) to obtain embeddings. A higher score reflects that the reasoning faithfully preserves both the order and semantic content of the input stream. In cases where multiple reasoning sentences correspond to one input sentence, we aggregate them as a single segment.

C.5 Similarity Map of Streaming CoT

For further qualitative evidence, we provide a sentence-level similarity map in Figure 1, which visualizes how reasoning aligns with the input across time. This map highlights diagonal patterns when the reasoning process maintains strong sequential consistency.

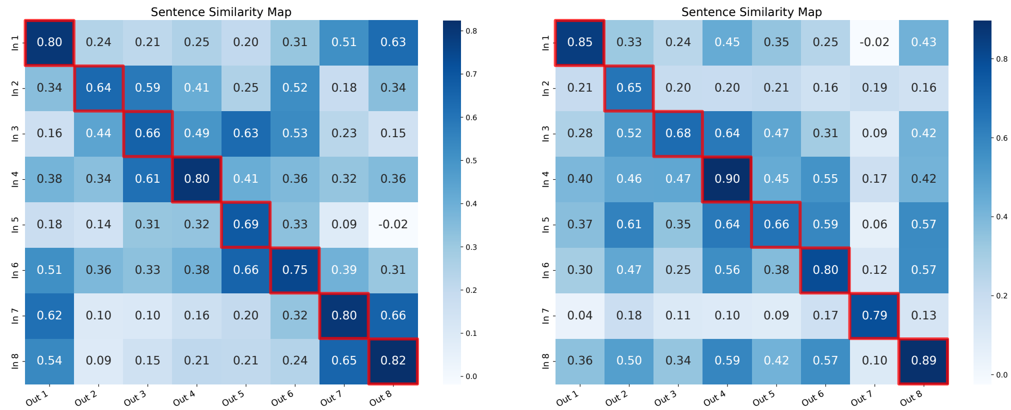

Figure 1: Examples of sentence-level similarity map between the input sentences and the reasoning content (from math reasoning task). High similarity along the diagonal demonstrates strong alignment, enabling evaluation through sequential consistency.

D Dataset Details

Math Reasoning We construct math reasoning data from GSM-Symbolic (Mirzadeh et al., 2025), MetaMathQA (Yu et al., 2024), and TuLu-personas-math-grade (Lambert et al., 2024).

GSM-Symbolic is a symbolic variant of GSM8K that reformulates arithmetic word problems into semi-structured symbolic expressions, explicitly preserving quantities, operators, and logical relations. It consists of two subsets, GSM-Symbolic-P1 (5K samples) and GSM-Symbolic-P2 (2.5K samples), where P2 includes longer and more complex problems. We randomly sample 4K and 2K instances from P1 and P2 for training, respectively, while the remaining data are used for generating training instances.

MetaMathQA is an augmented dataset derived from the GSM8K and MATH training sets, containing 40K samples. It integrates symbolic reasoning with natural language explanation, covering multi-step arithmetic and algebraic reasoning tasks. We randomly select 1K samples for testing and use the remaining data for training instance generation.

TuLu-personas-math-grade comprises 50K math-related examples generated by GPT-4o under diverse instructional and persona conditions. The dataset emphasizes reasoning diversity and linguistic variation, providing rich supervision signals for reasoning style adaptation. All samples are used for generating training data.

To align with the streaming objective of "thinking while reading," we retain inputs with at least three sentences and exclude tasks with very short inputs (e.g., proofs), which emphasize deep reasoning rather than streaming reasoning. We mix the generated data from all sources, and after filtering for correctness and evaluating the quality of streaming CoT, we retain 7.9K samples for training.

Logical Reasoning For logical reasoning, we evaluate on ProofWriter (Tafjord et al., 2020) and LogicNLI (Tian et al., 2021). ProofWriter is a benchmark designed for evaluating multi-step logical reasoning and theorem proving capabilities in natural language. It is derived from the EntailmentBank corpus, where each example consists of a hypothesis and a set of supporting facts expressed in natural language. The task requires the model to infer whether the hypothesis is entailed, contradicted, or neutral given the supporting facts, while optionally generating an explicit reasoning chain that connects the premises to the conclusion. LogicNLI is a natural language inference (NLI) benchmark designed to evaluate the logical reasoning ability of language models beyond surface-level semantics. Each example in LogicNLI consists of a premise and a hypothesis pair, annotated with one of three logical relations: entailment, contradiction, or neutral. Unlike conventional NLI datasets that focus primarily on lexical or syntactic cues, LogicNLI emphasizes reasoning over formal logic structures such as conjunction, disjunction, negation, quantifiers, and implication.

The ProofWriter dataset contains approximately 40K samples. We randomly sample 1K instances for testing, while the remaining data are used for generating training instances. For LogicNLI, we use its original 1k test set, while the remaining 16K training data is used for generating streaming CoT data. After applying the streaming CoT generation process, the resulting dataset contains approximately 20k instances for training.

Context-based QA Reasoning For context-based QA reasoning task, we evaluate StreamingThinker on PubMedQA (Jin et al., 2019) and HotpotQA (Yang et al., 2018) datasets.

PubMedQA is a biomedical question answering benchmark designed to evaluate factual reasoning and evidence-based inference in scientific texts. Each instance consists of a biomedical research question, a context paragraph extracted from PubMed abstracts, and a short-form answer annotated as yes, no, or maybe. The dataset emphasizes reasoning over factual statements, experimental findings, and logical connections.

HotpotQA is a large-scale question answering dataset designed to evaluate multi-hop reasoning across multiple supporting documents. Each example consists of a question, a set of supporting paragraphs from Wikipedia, and a corresponding answer. Unlike single-hop QA datasets that require reasoning within a single sentence or passage, HotpotQA explicitly requires models to integrate information from multiple contexts to derive the correct answer.

The PubMedQA dataset contains 1K samples for testing and 61K samples for training, where all of training samples are used for streaming CoT generation. The HotpotQA dataset includes 90K samples. We randomly sample 1K examples for testing and 60K for streaming CoT generation. After correctness filtering and streaming CoT quality evaluation, we retain a total of 16K samples for training.

E Model Details

E.1 Training Details

During training, we adopt a streaming mask region on top of the standard decoder-only LLM causal mask to enforce strict streaming constraints. Specifically, each training instance is composed of grouped *input* and *reasoning* segments, where the streaming mask (Eq. 2) ensures that reasoning at time t cannot attend to future inputs. Additionally, *the position encoding* of both the input and CoT is modified to group-wise streaming encoding, preventing positional conflicts.

For training, we set the batch size to 16 and employ the AdamW optimizer with a learning rate of 1e-5. We additionally enable activation checkpointing to mitigate memory overhead from maintaining parallel caches during training. These choices align the training setup with prior LLM pretraining while introducing only the modifications necessary for streaming compatibility.

E.2 Decoding Strategy

During inference, decoding must also respect the streaming paradigm. Unlike standard autoregressive generation where all inputs are visible, our decoding operates in a *streaming-aware* manner. Input tokens are continuously appended to a dedicated input cache, while reasoning tokens are generated in parallel using a separate reasoning cache. At each step, the streaming mask ensures that reasoning about the current sentence depends only on past inputs and past reasoning, never on unseen future inputs.

For generation, we follow the sampling parameters of Qwen3. This balances fluency and diversity while maintaining consistency across sequentially produced reasoning sentences. Importantly, because input and reasoning caches are maintained independently, decoding proceeds with lower latency: new inputs can be consumed without flushing or rewriting the reasoning cache, and reasoning updates can be emitted as soon as the corresponding input sentence completes. This strategy preserves alignment with streaming training while ensuring responsiveness in real-time scenarios. The streaming thinking decoding process is shown in Algorithm 1.

Algorithm 1 Streaming thinking with parallel KV caches

Input: Source length list S, target length list T.

Initialize input KV cache I_{cache}, output KV cache O_{cache}, and merged KV cache M_{cache}.
Read the first input sentence, and save hidden state to input KV cache I_{cache}.
`[Parallel] For Input KV Cache (prefill):`
while Input sentence is arrival **do**:
  Separate M_{cache} to I_{cache} and O_{cache}.
  Read the input sentence, and save hidden state to input KV cache I_{cache}.
end while
`[Parallel] For Output KV Cache (decoding):`
while Input sentence is arrival **do**:
  if $N_{EOS} \geq N_{EOT} + 1$: Sentence arrival time is less than LLM decoding time.
    Select a slice of the input KV cache I'_{cache} to keep $N_{EOS} == N_{EOT} + 1$.
  elif $N_{EOS} < N_{EOT} + 1$: Sentence arrival time exceeds LLM decoding time.
    Wait for the input KV cache prefilling and keep $N_{EOS} == N_{EOT} + 1$.
    Select a slice of the input KV cache I'_{cache}.
  Merge I'_{cache} and O_{cache} to M_{cache}.
  Decode the streaming thinking tokens and save to M_{cache}.
end while
`[End of Parallel]`
Generate controllable deep thinking with thinking intervention.
Return: The streaming CoT.

E.3 Relation between Parallel KV Caches and Prefill-Decode Separation

While our parallel KV caches mechanism may formally resemble techniques for Prefill-Decode separation (Zhong et al., 2024; Qin et al., 2024), the foundational motivations and operational goals behind the two approaches are fundamentally distinct.

Prefill-Decode separation is primarily a system-level optimization designed to enhance throughput in batched inference. It bifurcates the generation process into two discrete phases: a computationally intensive prefill stage that processes the input prompt in a parallel batch, and a memory-bandwidth-bound decoding stage for auto-regressive token generation. The core objective is to maximize hardware utilization by applying specialized computational kernels to each distinct phase, thereby improving efficiency for a fundamentally sequential task.

In contrast, the design of our parallel KV caches is driven by the goal of enabling the model to generate output concurrently while still consuming input. In other words, our approach is designed to overlap the processing of incoming data with the generation of the output, breaking the strict sequential dependency where generation can only begin after the entire input has been processed.

F Evaluation Metric

F.1 Reasoning Accuracy

Pass@1 Accuracy This metric measures the proportion of test instances in which the model's top-1 generated output exactly matches the ground-truth answer. Unlike relaxed metrics that consider multiple sampled generations (e.g., pass@k), pass@1 provides a strict estimate of the model's reliability under single-shot decoding, which is the default usage setting for most real-world applications. A higher pass@1 score thus indicates stronger reasoning consistency and correctness without relying on sampling diversity.

F.2 Reasoning Latency

Token-length Delay In streaming tasks, token-length delay measures *how many tokens must be consumed before the model begins its reasoning process*. Formally, it is defined as the gap between the start of the input stream and the position of the first reasoning token. A smaller token delay indicates that the model can initiate reasoning earlier in the stream, leading to faster overlap between input consumption and output generation. This metric thus reflects the responsiveness of reasoning onset in the streaming paradigm.

$$\text{Token-Delay} = \text{Position of first reasoning token} \tag{1}$$

Time Delay Time delay measures the real-time latency *between the arrival of the last input token and the emission of the first answer token*. This metric captures system-level responsiveness, incorporating factors such as decoding speed, cache management, and hardware throughput. While token delay evaluates the generation behavior in discrete units, time delay provides a wall-clock perspective that better reflects the user's perceived waiting time. The Figure 2 illustrates the latency comparison between streaming thinking and batch thinking.

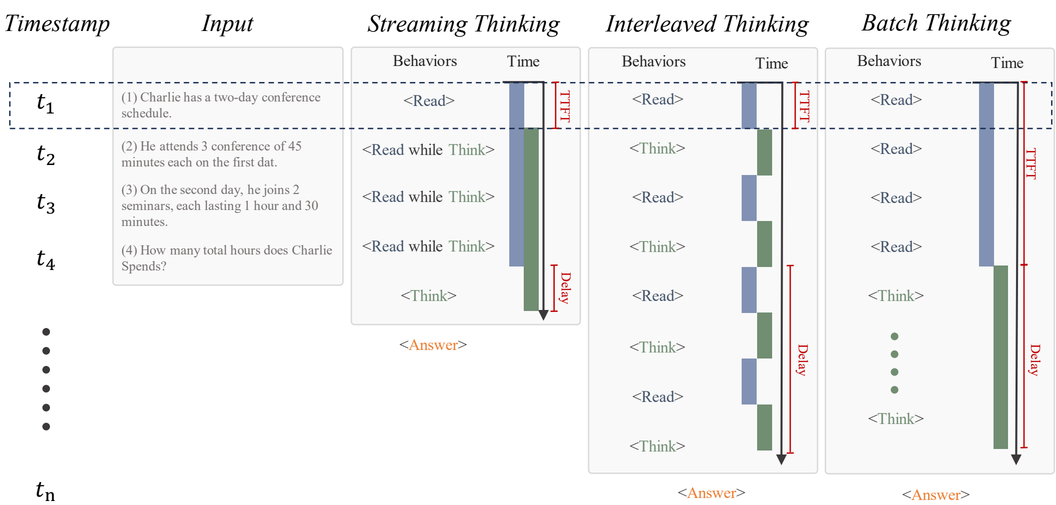

Figure 2: Comparison of reasoning paradigms. Streaming thinking enables concurrent reading and reasoning with minimal delay, while interleaved thinking alternates between the two but still accumulates overhead, and batch thinking postpones reasoning until all inputs are read, leading to the largest delay.

G The Use of Large Language Models (LLMs)

LLMs are only used for language refinement (e.g., grammar and style corrections). They are not involved in research conception, execution, or analysis.