Core Context Aware Transformers for Long Context Language Modeling

Yaofo Chen*, Zeng You*, Shuhai Zhang*, Haokun Li, Yirui Li, Yaowei Wang†, Mingkui Tan†

Background

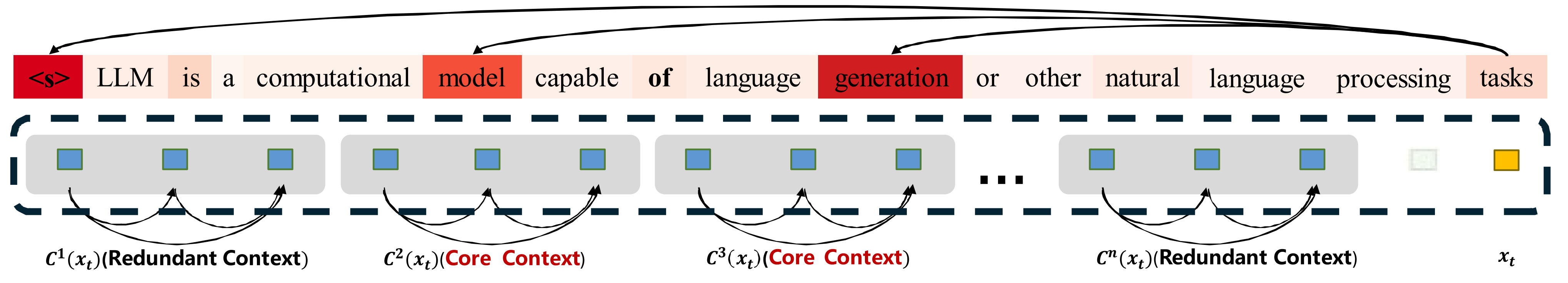
- **Problem**: Self-attention in LLMs suffers from quadratic computational complexity and redundant token interactions in long sequences (e.g., 128K tokens), leading to inefficiency.
- **Limitations of Existing Methods**: Prior approaches either discard tokens or rely on predefined sparse patterns, compromising context completeness and adaptability.

We aim to design a **lightweight, plug-and-play** attention mechanism that **dynamically** compresses redundant tokens while preserving both global dependencies and local fine-grained context .

Core Context vs. Redundancy in Long Context Modeling

- **Core Context**: Tokens with high semantic relevance to the target token x_t (e.g., $C^2(x_t)$), critical for accurate long context modeling.
- **Redundancy Context**: Tokens with weak semantic connections (e.g., $C^1(x_t)$ and $C^n(x_t)$), introducing unnecessary computational overhead.

Traditional attention mechanisms assign disproportionately high scores to core contexts while retaining redundant tokens, leading to inefficiency in long-sequence modeling.



<s> LLM is a computational model capable of language generation or other natural language processing tasks
$C^1(x_t)$(Redundant Context)
$C^2(x_t)$(Core Context)
$C^3(x_t)$(Core Context)
$C^n(x_t)$(Redundant Context)
x_t

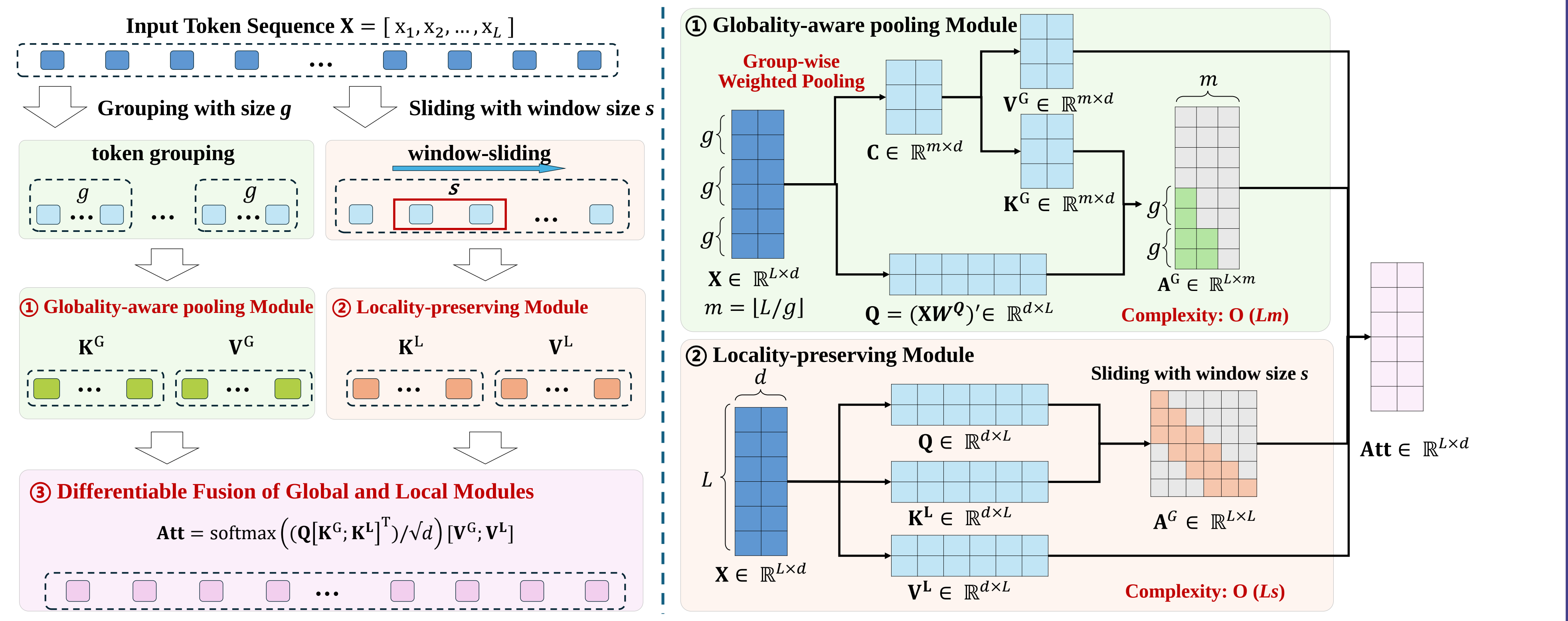
Our Goal: **1)** preserve core contexts for accurate attention computation; **2)** compress the redundant information within the long context to reduce computational cost.

Contributions

- We propose a **plug-and-play Core Context Aware Attention** for efficient long-context modeling by taking a set of core tokens as efficient proxies for attention. Unlike traditional efficient attention methods that require extensive retraining, our CCA-Attention can be easily integrated into pretrained LLMs with **minimal fine-tuning effort**.
- We develop a **dynamic globality-aware pooling module** that adaptively derives core tokens based on their importance. By compressing input tokens into core tokens, our method captures the most relevant global context, leading to more accurate and effective long-term dependency modeling than static or random token selection approaches.
- We achieve significant improvements compared with other baseline methods in both long-context modeling performance and computational efficiency, achieving a **7.9× speedup** and **93% KV cache memory reduction** compared to full self-attention when processing 128K token contexts, while outperforming existing efficient attention mechanisms in long-context modeling.

Overview of Core Context Aware Attention

Our CCA-Attention includes two components: 1) globality-aware pooling module encapsulates the input tokens into core tokens according to the importance, serving as representative proxies of input tokens for attention, 2) Locality-preserving module incorporates the local context from neighboring tokens, acting as supplement for the globality-aware pooling module.



Input Token Sequence $X = [x_1, x_2, ..., x_L]$
Grouping with size g
Sliding with window size s
token grouping
window-sliding
① Globality-aware pooling Module
② Locality-preserving Module
K^G V^G K^L V^L
③ Differentiable Fusion of Global and Local Modules
$Att = softmax((Q[K^G; K^L]^T)/\sqrt{d})[V^G; V^L]$
Group-wise Weighted Pooling
$C \in \mathbb{R}^{m\times d}$
$V^G \in \mathbb{R}^{m\times d}$
$K^G \in \mathbb{R}^{m\times d}$
$X \in \mathbb{R}^{L\times d}$
$m = \lfloor L/g \rfloor$
$Q = (XW^Q)' \in \mathbb{R}^{d\times L}$
$A^G \in \mathbb{R}^{L\times m}$
Complexity: O (Lm)
$Q \in \mathbb{R}^{d\times L}$
$K^L \in \mathbb{R}^{d\times L}$
$V^L \in \mathbb{R}^{d\times L}$
$A^G \in \mathbb{R}^{L\times L}$
Complexity: O (Ls)
$Att \in \mathbb{R}^{L\times d}$

Globality-aware Pooling Module

Insight: Computational resources can be dynamically allocated to core contexts.

Solution: We divide $\mathbf{X}{=}[\mathbf{x}_1;\ldots;\mathbf{x}_L]$ into $m{=}\lfloor L/g\rfloor$ groups and derive core token c_i for each group

$$\mathbf{c}_i{=}\text{softmax}\left(\frac{\mathbf{Q}_{ig}\mathbf{K}'^{\top}_{\mathcal{I}_i}}{\sqrt{d}}\right)\mathbf{X}^{\text{G}}_{\mathcal{I}_i}\in\mathbb{R}^{1\times d}, i=1,\ldots,m, \tag{1}$$

where $\mathbf{X}^{\text{G}}_{\mathcal{I}_i}{\in}\mathbb{R}^{g\times d}$ is the i-th group and $\mathbf{Q}_{ig}$ is the last query in the group. We adopt core tokens $\mathbf{C}{=}[\mathbf{c}_1;\mathbf{c}_2;\ldots;\mathbf{c}_m]$ to calculate the key and value matrices for the query $\mathbf{Q}_i$

$$\widetilde{\mathbf{K}}^{\text{G}}_{\mathcal{T}_i}{=}[\mathbf{K}^{\text{G}}_1;\cdots;\mathbf{K}^{\text{G}}_j],\ \widetilde{\mathbf{V}}^{\text{G}}_{\mathcal{T}_i}{=}[\mathbf{V}^{\text{G}}_1;\cdots;\mathbf{V}^{\text{G}}_j],\ \ \mathbf{K}^{\text{G}}{=}\mathbf{C}\mathbf{W}^K,\mathbf{V}^{\text{G}}{=}\mathbf{C}\mathbf{W}^V,\ \ j{=}\max(0,\lfloor(i{-}s)/g\rfloor) \tag{2}$$

Locality-preserving Module

Insight: Core tokens focus on coarse-grained global context, overlooking fine-grained local one.

Solution: We further adopt $s+((i{-}s) \mod g)$ local tokens for each query $\mathcal{Q}_i$ to

$$\widetilde{\mathbf{K}}^{\text{L}}_{\mathcal{U}_i}{=}[\mathbf{K}^{\text{L}}_k;\cdots;\mathbf{K}^{\text{L}}_i],\ \ \widetilde{\mathbf{V}}^{\text{L}}_{\mathcal{U}_i}{=}[\mathbf{V}^{\text{L}}_k;\cdots;\mathbf{V}^{\text{L}}_i],\quad k{=}\max(1,i{-}s{-}((i{-}s)\bmod g)) \tag{3}$$

Differentiable Fusion of Global and Local Modules

We concatenate the key and value matrices from both module to calculate the outputs for $\mathbf{Q}_i$

$$\mathbf{Att}_i=\text{softmax}\Big(\frac{\mathbf{Q}_i[\widetilde{\mathbf{K}}^{\text{G}}_{\mathcal{T}_i};\widetilde{\mathbf{K}}^{\text{L}}_{\mathcal{U}_i}]^{\top})}{\sqrt{d}}\Big)[\widetilde{\mathbf{V}}^{\text{G}}_{\mathcal{T}_i};\widetilde{\mathbf{V}}^{\text{L}}_{\mathcal{U}_i}]. \tag{4}$$

We represent the final output of our CCA-Attention as $\mathbf{Att}=[\mathbf{Att}_1;\mathbf{Att}_2;\ldots;\mathbf{Att}_L]$.

Comparisons with SoTA Methods

- **Comparisons on LongBench-E. We report the latency and memory footprint of LLaMA2-7B-32K and LLaMA2-7B-80K with 32K and 64K contexts on A800 GPUs, respectively.**

Methods	S. QA	M. QA	Sum.	FS. Learning	Synthetic	Code	Avg.	FTL (s)	Mem. (GB)
LLaMA2-7B-32K (Vanilla Self-Attention)	2.75	1.85	**12.43**	**66.28**	0.34	48.99	**22.11**	9.15	35.58
StreamingLLM	**4.75**	2.94	2.97	48.20	0.66	30.16	14.95	5.75 (1.6×)	22.94 (35%↓)
LM-Infinite	2.04	2.33	1.98	57.45	0.3	48.46	18.76	4.72 (1.9×)	26.35 (26%↓)
MInference	3.68	3.05	10.97	66.26	0.61	42.30	21.14	4.20 (2.2×)	33.52 (6%↓)
CCA-LLM (Ours)	3.63	**3.98**	7.79	61.79	**2.64**	**51.36**	21.86	**2.59 (3.5×)**	**19.12 (46%↓)**
LLaMA2-7B-80K (Vanilla Self-Attention)	3.22	2.71	3.90	**64.98**	0.56	**59.16**	**22.42**	32.43	60.03
StreamingLLM	2.07	2.32	0.37	45.03	**2.67**	37.17	14.94	9.04 (3.6×)	37.45 (37%↓)
LM-Infinite	2.54	1.53	2.22	61.29	1.08	58.54	21.20	8.27 (3.9×)	41.54 (31%↓)
MInference	2.44	3.49	4.41	64.26	0.28	57.60	22.08	8.14 (4.0×)	54.09 (10%↓)
CCA-LLM (Ours)	**5.62**	**4.34**	**8.99**	59.60	0.48	54.40	22.24	**6.42 (5.7×)**	**33.86 (44%↓)**

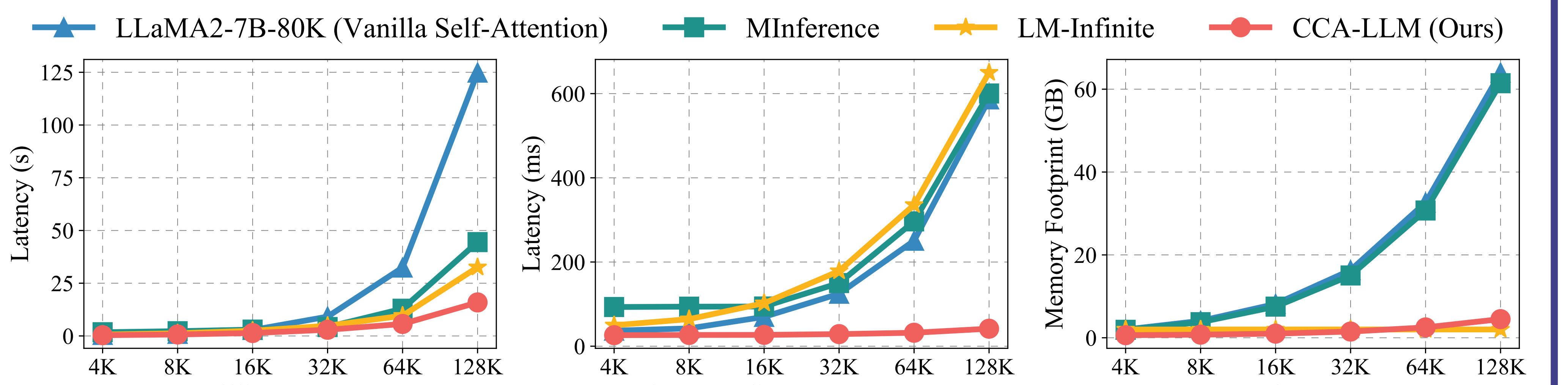
- **Comparisons on LongBench-E on the latest model.**

Methods	S. QA	M. QA	Sum.	FS. Learning	Synthetic	Code	Avg.	FTL (s)	Mem. (GB)
LLaMA3.1-8B-128K (Vanilla Self-Attention)	16.71	10.75	20.32	**68.75**	**48.93**	62.10	**37.93**	9.55	40.38
MInference	16.33	10.71	**20.44**	68.41	48.06	**62.50**	37.74	4.93 (1.9×)	35.95 (11%↓)
CCA-LLM (Ours)	**17.90**	**16.41**	19.63	67.20	43.76	61.98	37.81	**3.08 (3.1×)**	**20.63 (49%↓)**
Qwen2.5-7B-128K (Vanilla Self-Attention)	16.67	**18.18**	**18.70**	66.81	45.34	**64.56**	**38.38**	10.58	35.11
MInference	16.20	17.21	18.59	**67.10**	38.28	62.95	36.72	4.86 (2.2×)	32.40 (8%↓)
CCA-LLM (Ours)	**16.91**	17.07	18.60	66.89	**45.50**	63.52	38.08	**2.74 (3.9×)**	**19.31 (45%↓)**

- **Comparisons of different models on multi-document EM score evaluated at lengths from 4K to 128K. We report the latency within a context of 128K on two A800 GPUs.**

Methods	4K	8K	16K	32K	64K	128K	Avg.	FTL (s)
LLaMA2-7B-80K (Vanilla Self-Attention)	**39.4**	**37.8**	**37.6**	**36.2**	34.6	30.3	**36.0**	124.85
StreamingLLM	33.6	26.0	32.2	30.6	27.4	25.1	29.2	34.74 (3.6×)
LM-Infinite	31.6	25.6	32.4	32.2	28.2	26.3	29.4	32.57 (3.8×)
MInference	39.0	32.4	37.4	36.0	32.3	28.9	34.3	20.18 (6.2×)
CCA-LLM (Ours)	39.3	33.2	35.4	31.4	**35.3**	**32.0**	34.4	**15.89 (7.9×)**

- **Comparisons in terms of both computational and storage overhead on LLaMA2-7B-80K.**



(a) Pre-filling Latency (FTL) (b) Decoding Latency (ITL) (c) KV Cache Memory Usage

Contact Information and Code

- Email: chenyaofo@gmail.com
- Email: mingkuitan@scut.edu.cn
- Code: https://github.com/chenyaofo/CCA-Attention

Paper Code