Efficient Streaming Language Models with Attention Sinks

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, Mike Lewis

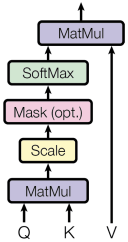
MIT, Meta AI, CMU

December 7, 2023

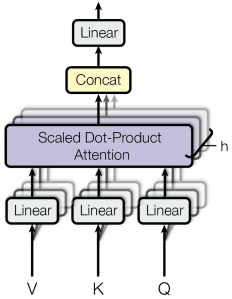
Presented by Huan Liang

What is attention?

Scaled Dot-Product Attention



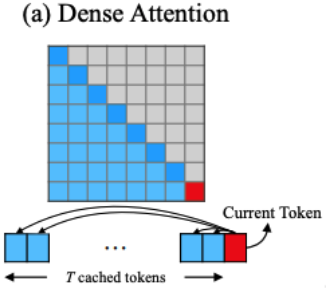
Multi-Head Attention



What if we want take a pre-trained LLM and let it run forever?

A couple of problems:

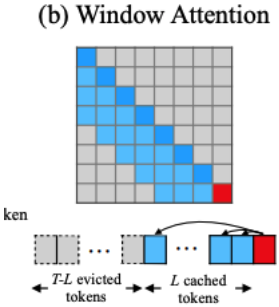
- Poor performance on sequences longer than pretrained lengths (4K for Llama-2)
- Poor $O(T^2)$ time complexity, unsustainable as sequences increase in lengths
- Memory issues with caching key and value states of previous tokens



So, what have people done?

Window Attention

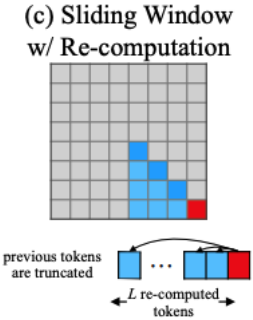
- Cache the most recent L tokens' KV
- Attend current token with those KVs
- Better time complexity, poor performance



So, what have people done?

Sliding Window Attention with Re-computation

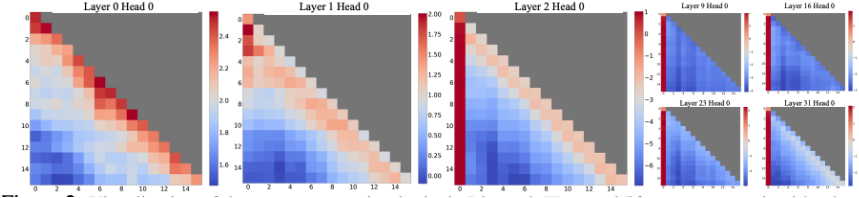
- Recompute attention values for most recent L tokens,
- Use recomputed values for current token
- Better performance, poor time complexity



Why doesn't window attention work?

Attention Sink!

- Large amount of attention scores allocated to initial tokens
- Softmax operation, forces attention scores to sum to 1 (even if values are small)
- Why initial tokens?



Initial tokens are important

- In a autoregressive nature, initial tokens are visible to all subsequent tokens
- Absolute position matters more than semantics of first few tokens
- First four tokens is all you need

Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

Cache Config	0+2048	1+2047	2+2046	4+2044	8+2040
Falcon-7B	17.90	12.12	12.12	12.12	12.12
MPT-7B	460.29	14.99	15.00	14.99	14.98
Pythia-12B	21.62	11.95	12.09	12.09	12.02

Cache Config	0+4096	1+4095	2+4094	4+4092	8+4088
Llama-2-7B	3359.95	11.88	10.51	9.59	9.54

Learnable "sink" token

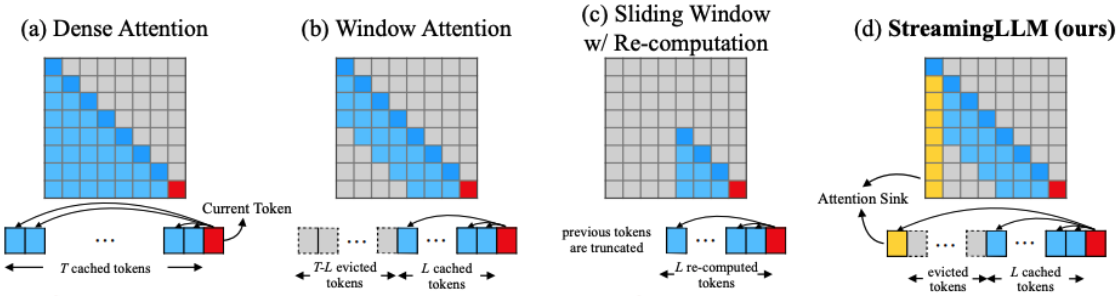
- One learnable sink token prepended helps

Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02

Positional encodings

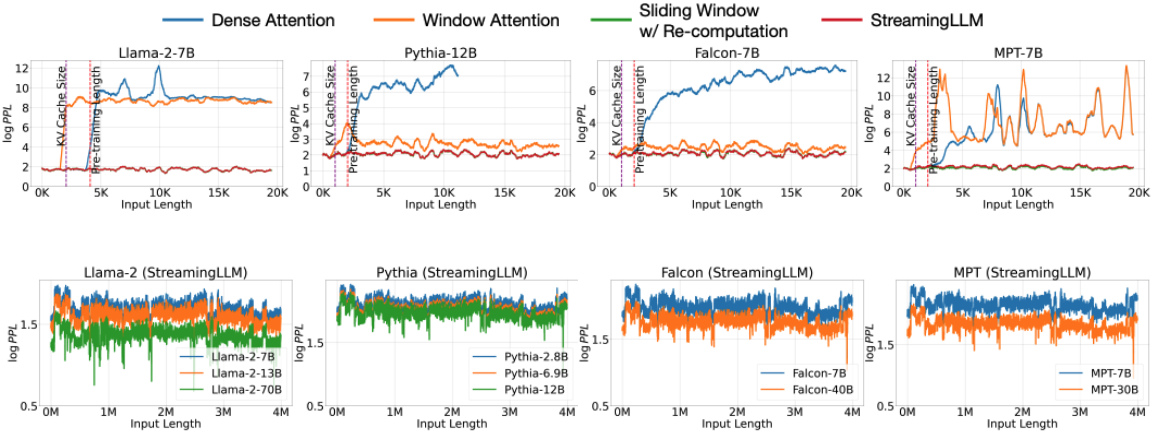
- StreamingLLM focuses on positions within the cache rather than those in the original text. This distinction is crucial for StreamingLLM's performance. For instance, if the current cache (Figure 4) has tokens [0, 1, 2, 3, 6, 7, 8] and is in the process of decoding the 9th token, the positions assigned are [0, 1, 2, 3, 4, 5, 6, 7], rather than the positions in the original text, which would be [0, 1, 2, 3, 6, 7, 8, 9].

StreamingLLM



Language modeling on long texts across LLM Families and Scales

- PG19 test set (100 long books)
- Llama-2 with 2048 cache, Falcon, Pythia, MPT with 1024 cache



Results of pre-training with sink token

- Two 160M parameter language models, Pythia-160M codebased, 8xA6000 GPUs, Pile dataset
- One with Sink Token, one without

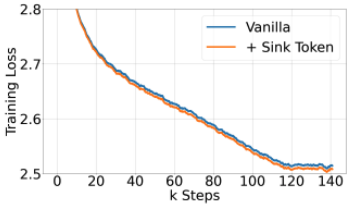
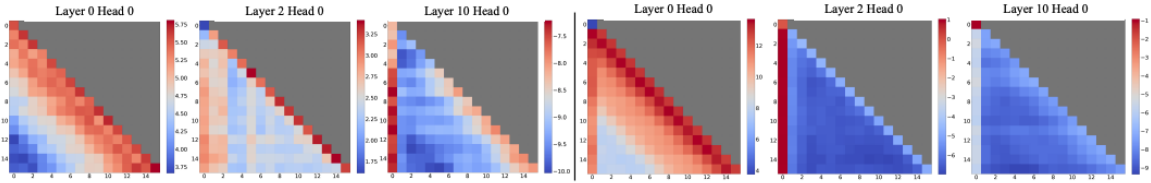


Figure 6: Pre-training loss curves of models w/ and w/o sink tokens. Two models have a similar convergence trend.

Table 4: Zero-shot accuracy (in %) across 7 NLP benchmarks, including ARC-[Challenge, Easy], HellaSwag, LAMBADA, OpenbookQA, PIQA, and Winogrande. The inclusion of a sink token during pre-training doesn't harm the model performance.

Methods	ARC-c	ARC-e	HS	LBD	OBQA	PIQA	WG
Vanilla	18.6	45.2	29.4	39.6	16.0	62.2	50.1
+Sink Token	**19.6**	**45.6**	**29.8**	**39.9**	**16.6**	**62.6**	**50.8**

Attention visualization



Pre-Trained without Sink Token

Pre-Trained with Sink Token

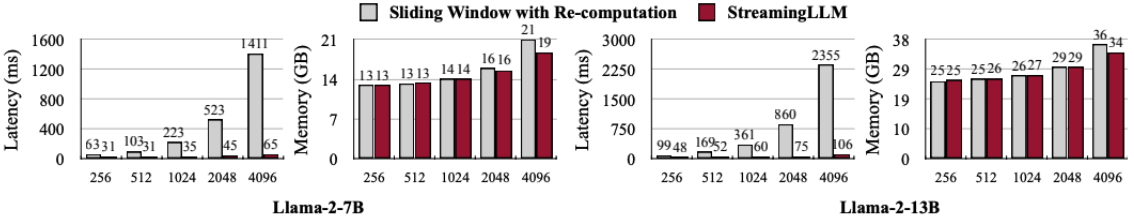
Results on streaming question answering with instruction-tuned models

Table 5: Accuracy (in %) on the ARC-[Easy, Challenge] datasets. Questions were concatenated and answered in a streaming manner to mimic a real-world chat setting. The dense baseline fails due to Out-of-Memory (OOM) errors. Window attention has poor accuracy. StreamingLLM has comparable results with the one-shot sample-by-sample baseline. Window attention and StreamingLLM use cache sizes of 1024.

Model	Llama-2-7B-Chat		Llama-2-13B-Chat		Llama-2-70B-Chat	
Dataset	Arc-E	Arc-C	Arc-E	Arc-C	Arc-E	Arc-C
One-shot	71.25	53.16	78.16	63.31	91.29	78.50
Dense	OOM					
Window	3.58	1.39	0.25	0.34	0.12	0.32
StreamingLLM	71.34	55.03	80.89	65.61	91.37	80.20

Efficiency results and cache sizes

Figure 9: Performance on the StreamEval benchmark. Accuracies are averaged over 100 samples.



Cache	4+252	4+508	4+1020	4+2044
Falcon-7B	13.61	12.84	**12.34**	12.84
MPT-7B	**14.12**	14.25	14.33	14.99
Pythia-12B	13.17	12.52	**12.08**	12.09

Cache	4+508	4+1020	4+2044	4+4092