Efficient Streaming Language Models with Attention Sinks

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

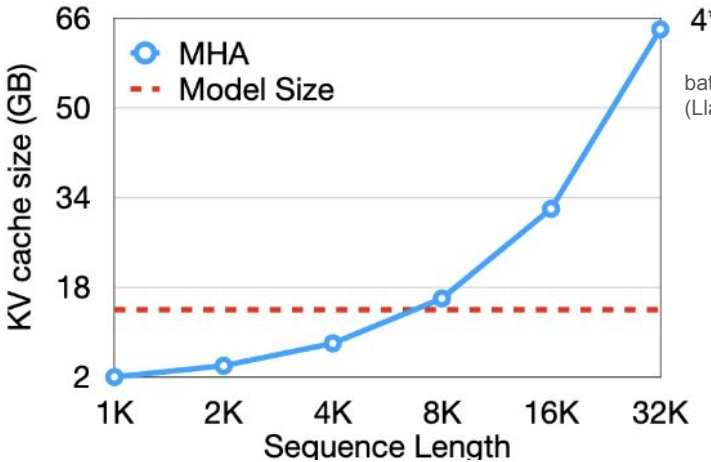
CWRU

Presented by Ethan Fang

Background

The problem of long context: large KV cache

- we know previously that during decoding we need to store the KV pairs of all previous tokens so that we can perform attention computation



4*32*32*128*32K*2*2=**64GB**

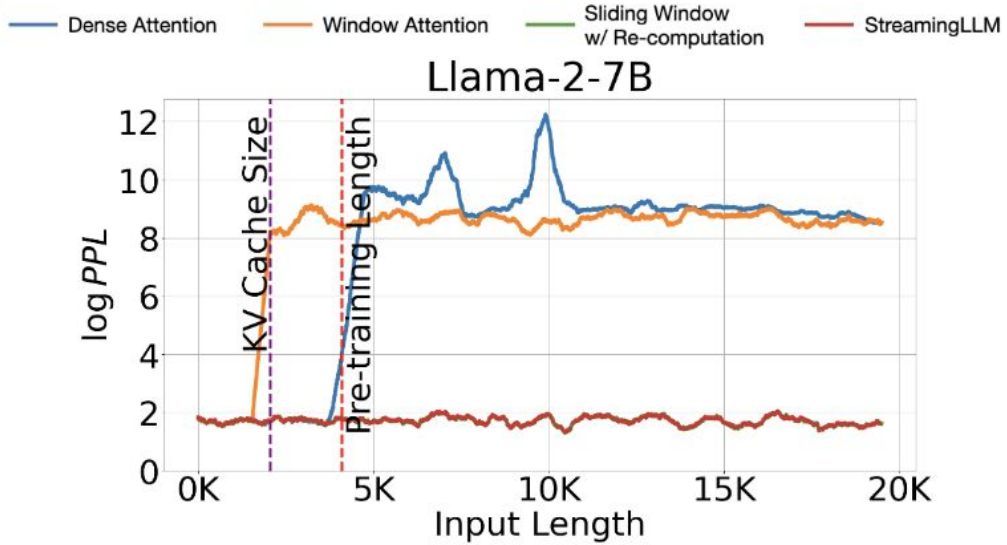
batchsize*layers*kvheads*n_{end}*length*K&V*FP16
(Llama-2-7B)

CWRU

Current Methods

Window Attention - caching only the most recent KV states

- issue arises: model collapses when the text length surpasses the cache size



CWRU

Investigation -> Discovery

- designed a series of experiments to understand why window attention performance dropped
- looked at attention score distributions in autoregressive LLMs
  - surprisingly large amount of attention was consistently allocated to initial tokens, even when those tokens were far from the current token being processed and not semantically important
- Visualized attention maps and revealed that attention was disproportionately focused on initial tokens

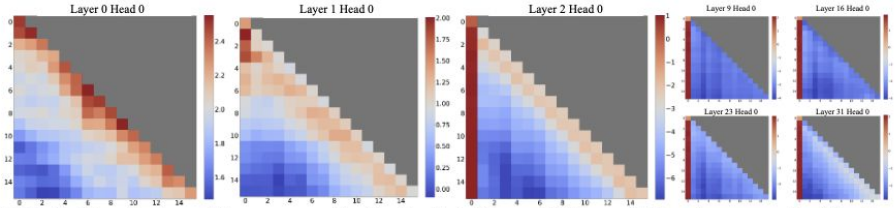


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two layers, the model heavily attends to the initial token across all layers and heads.

CWRU

What is an Attention Sink?

Phenomenon where initial tokens in a sequence attract disproportionately high attention scores from the model, even if they are not semantically important

- “anchor” the attention mechanism, stabilizing the model’s focus during long sequence generation

CWRU

Why does the Attention Sink exist?

Behavior is caused by the softmax function

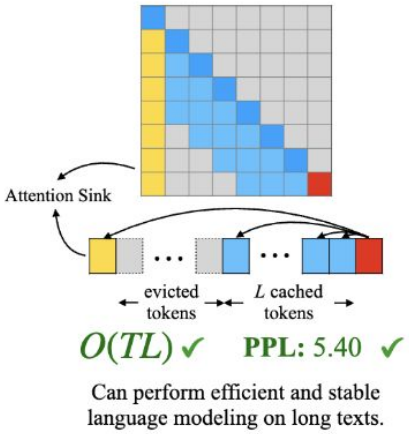
- self attention computes attention scores by comparing the query with KV pairs from previous tokens
- since initial tokens are always present in the sequence and have been seen by all the tokens generated, the softmax mechanism tends to assign excess attention to them, even if not semantically important

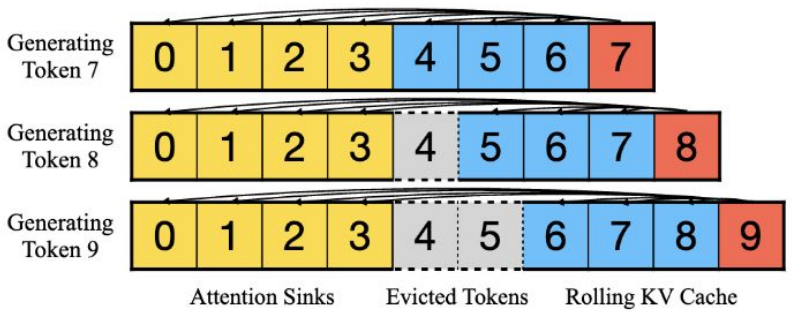
Softmax normalizes attention scores, tends to distribute attention to tokens that have been visible for the longest time.

CWRU

Introducing… StreamingLLM

- Enables LLMs trained with finite attention window to handle infinite text lengths without additional training
- Alongside current sliding window tokens, reintroduce few starting tokens' KV in attention computation





CWRU

Advantages

Memory efficiency

- Caching initial tokens + most recent = less tokens in memory

Stable performance

- provides stable perplexity across long sequences, overcoming performance degradation seen with window attention

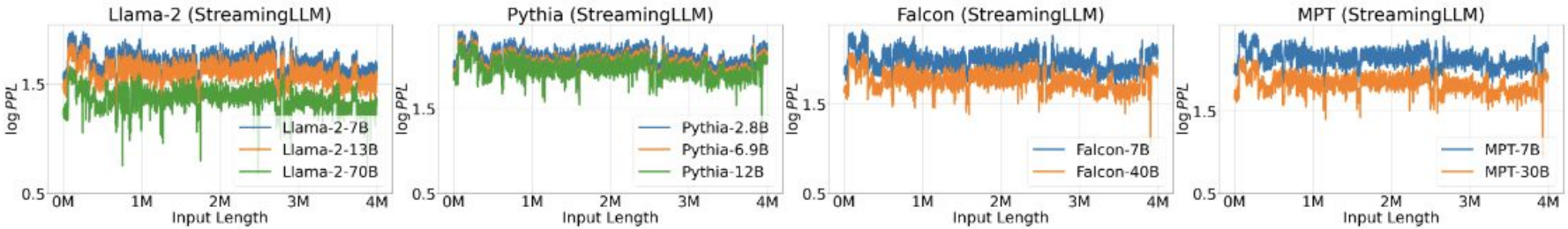
Improved decoding speed

- sliding window has to recompute attention scores for every token each time it is processed
- Model handles new tokens without having to revisit entire sequence, allowing it to process long sequences much faster

CWRU

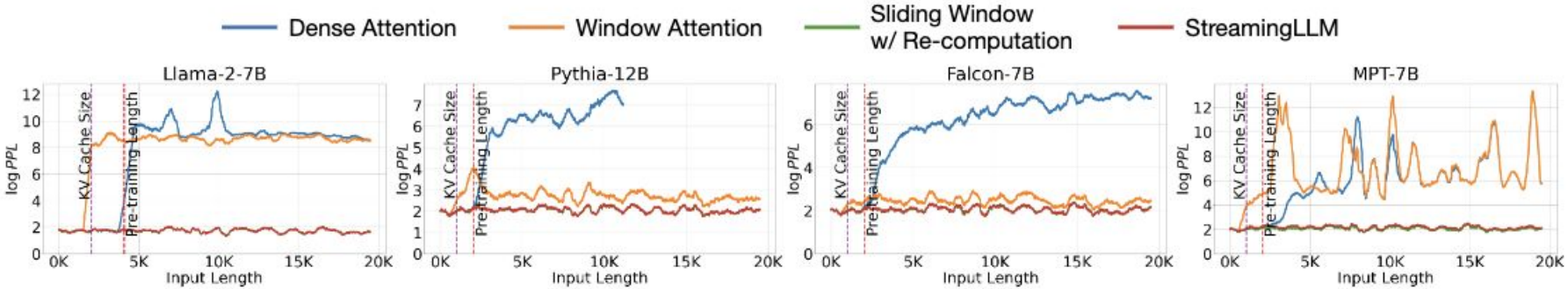
Performance analysis

Model families shown can now model up to 4 million tokens



CWRU

Performance analysis



CWRU

Latency Analysis

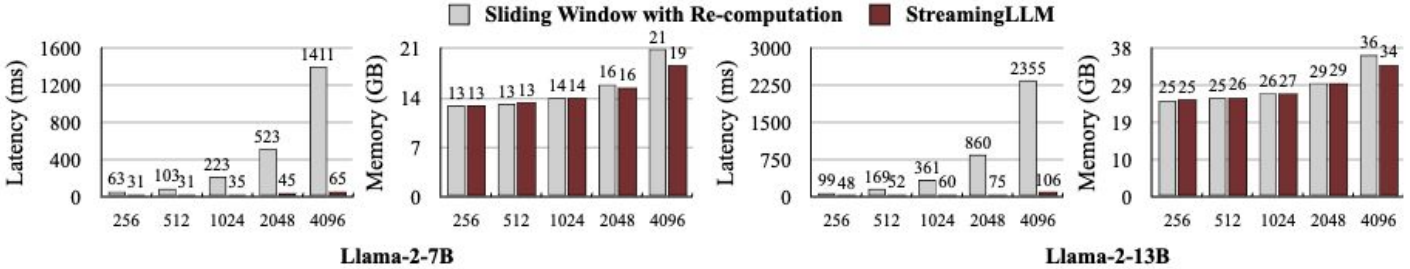


Figure 10: Comparison of per-token decoding latency and memory usage between the sliding window approach with re-computation baseline and StreamingLLM, plotted against the cache size (attention window size) on the X-axis. StreamingLLM delivers a remarkable speedup of up to 22.2× per token and retains a memory footprint similar to the re-computation baseline.

CWRU

How many initial tokens do we need?

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

CWRU

4 tokens to 1 token – Pretraining with Sink token

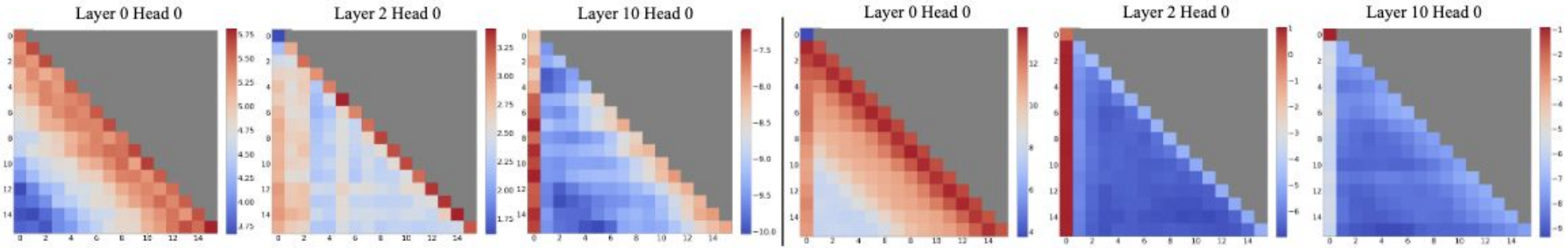
Authors suggest introducing special “sink token” at the start of every input sequence as a learnable parameter

- model would assign excess attention to the token
- learnable token, such that it would adapt during pre-training to absorb unnecessary attention

Reduces the number of tokens that need to be cached in memory

- makes attention distribution more controlled and consistent across different tasks and sequences

CWRU



Pre-Trained without Sink Token

Pre-Trained with Sink Token

CWRU

No Loss of Accuracy

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**

CWRU

Performance analysis

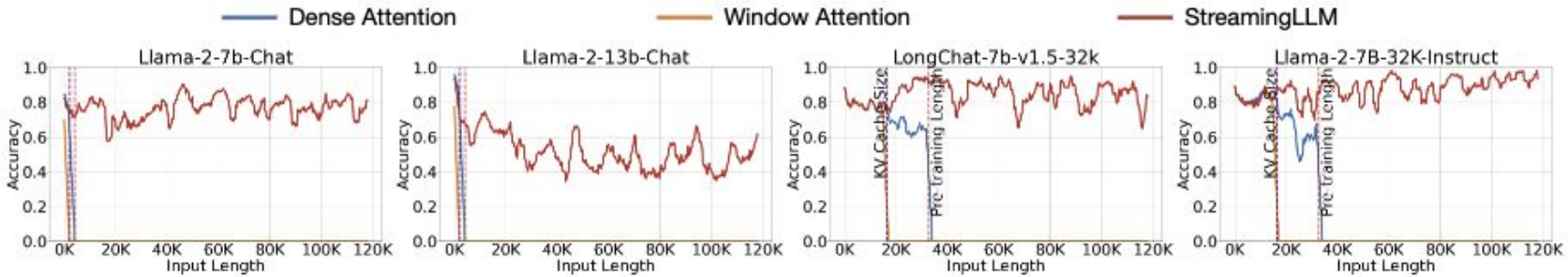
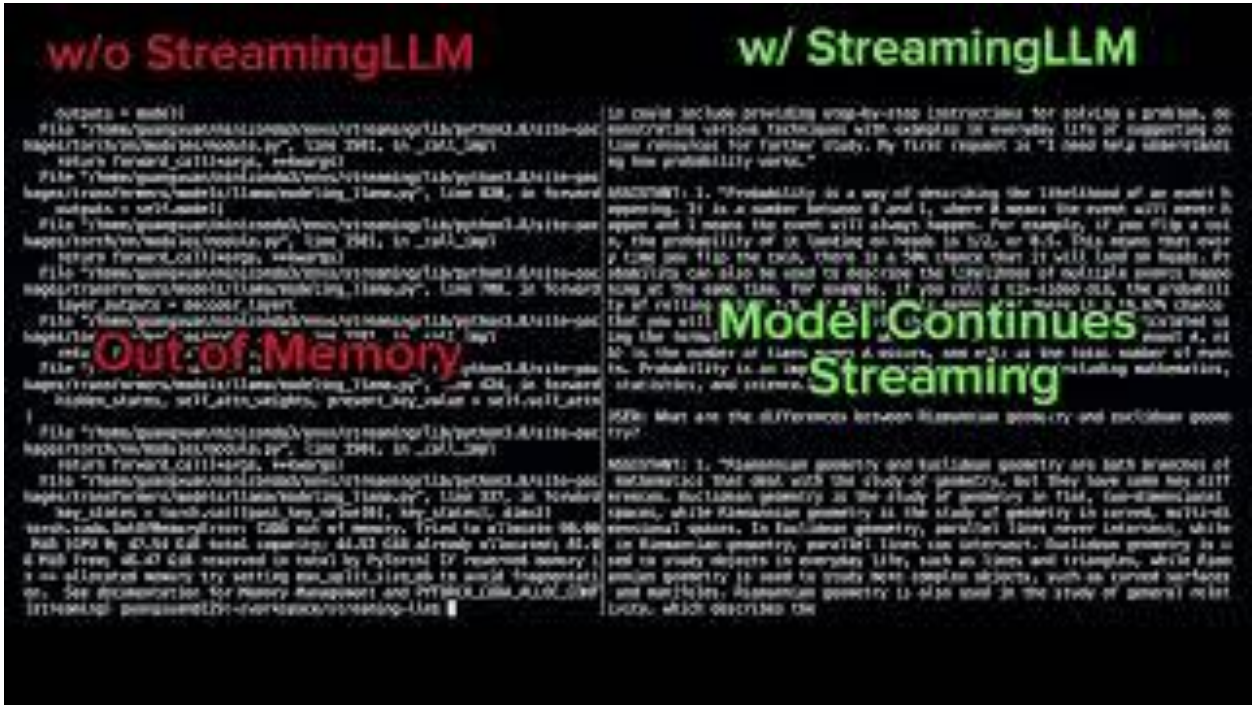


Figure 9: Performance on the StreamEval benchmark. Accuracies are averaged over 100 samples.

CWRU



video via. mit-han-lab github

Future Work

- Improving long term memory mechanisms, how to handle tasks that require persistent context over very long sequences or interactions

CWRU

Thank you!

CWRU