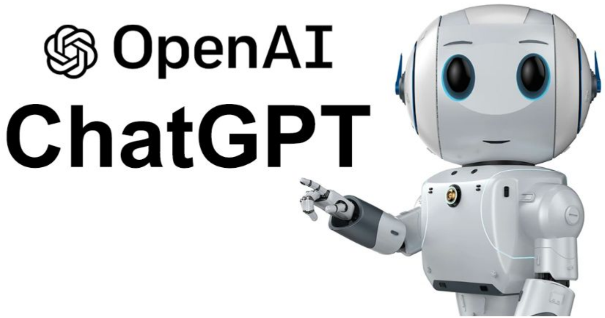
Efficient Streaming LMs with Attention Sinks

Yitong Chen, Ruihang Lai, Judy Jin, Nasrin Kalanat

11868 LLM System

Motivation: Use cases



You

In Linux, how can I modify a specific line of a file from pure command line without open any editor?

ChatGPT

You can use `sed` (stream editor) to modify a specific line of a file from the command line without opening any editor. Here's how you can do it:

bash — Copy code

```bash
sed -i 's/old_text/new_text/' file.txt
```

Replace `old_text` with the text you want to replace and `new_text` with the text you want to replace it with. `file.txt` is the file you want to modify.

You

I don't want to substitute. I just want to completely overwrite that line to new contents

ChatGPT

If you want to completely overwrite a specific line with new contents without using an editor, you can use `sed` with the `c` command. Here's how you can do it:

bash — Copy code

```bash
sed -i 'Nc\new_contents' file.txt
```

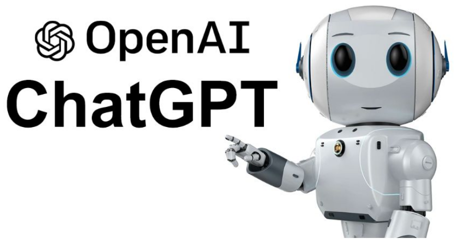
Replace `N` with the line number you want to overwrite and `new_contents` with the new text you want to replace the line with.

Carnegie Mellon University

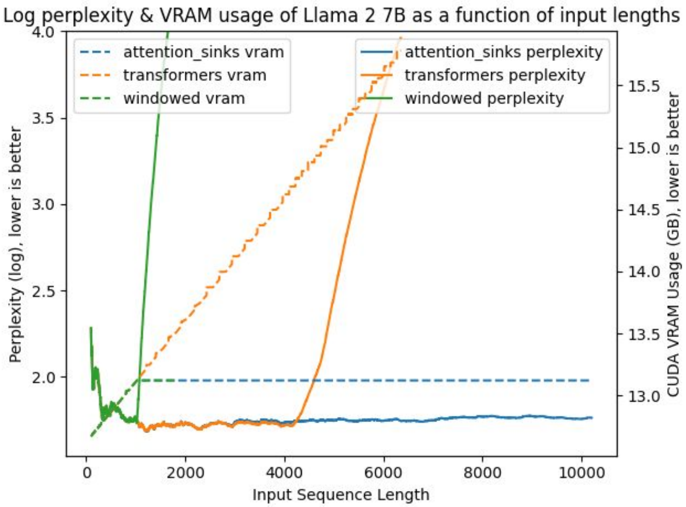
https://github.com/mit-han-lab/streaming-llm/blob/main/assets/StreamingLLM.pdf

Challenges of Deploying LLMs in Streaming Applications

- Urgent need for LLMs in streaming applications such as multi-round dialogues, where long interactions are needed.



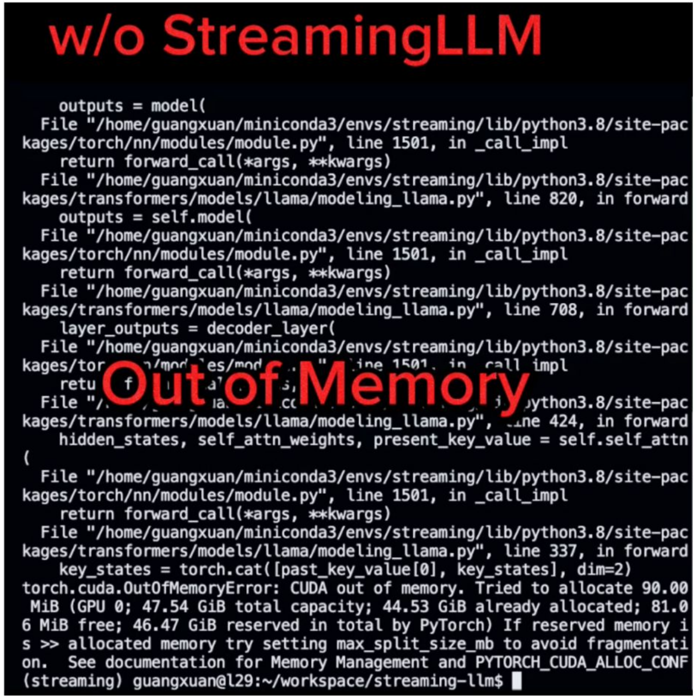
- Challenges
  - Extensive memory consumption during the decoding stage.
  - Inability of popular LLMs to generalize to longer text sequences.



https://github.com/tomaarsen/attention_sinks

https://github.com/mit-han-lab/streaming-llm/blob/main/assets/StreamingLLM.pdf

Challenges of Deploying LLMs in Streaming Applications



Carnegie Mellon University

https://github.com/mit-han-lab/streaming-llm/blob/main/assets/StreamingLLM.pdf

Directions of Approaching the Problem?

Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System

Directions of Approaching the Problem?

Applying LLMs with fixed context size on longer content

Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System

Relative Rotary Positional Embedding

https://medium.com/ai-insights-cobet/rotary-positional-embeddings-a-detailed-look-and-comprehensive-understanding-4ff66a874d83

Directions of Approaching the Problem?

Enabling LLMs' context sizes to be larger

Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System

Position Interpolation

[0, 1, 2, 3, …, 8191]

divided by 2

[0.0, 0.5, 1.0, 1.5, …, 4095.5]

Directions of Approaching the Problem?

Enabling LLMs' context sizes to be larger

Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System

FlashAttention

Zhitong Guo, Xinran Wan, Haoze He, Alicia Sui

Directions of Approaching the Problem?

Making more use of your context

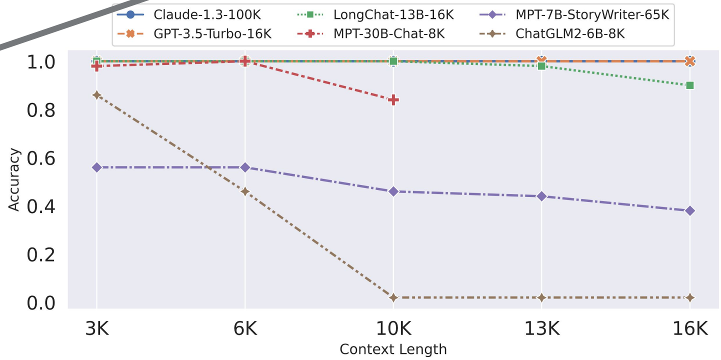
Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System



Carnegie Mellon University

https://lmsys.org/blog/2023-06-29-longchat/

Directions of Approaching the Problem?

Applying LLMs with fixed context size on longer content

Length extrapolation

Context Window Extension

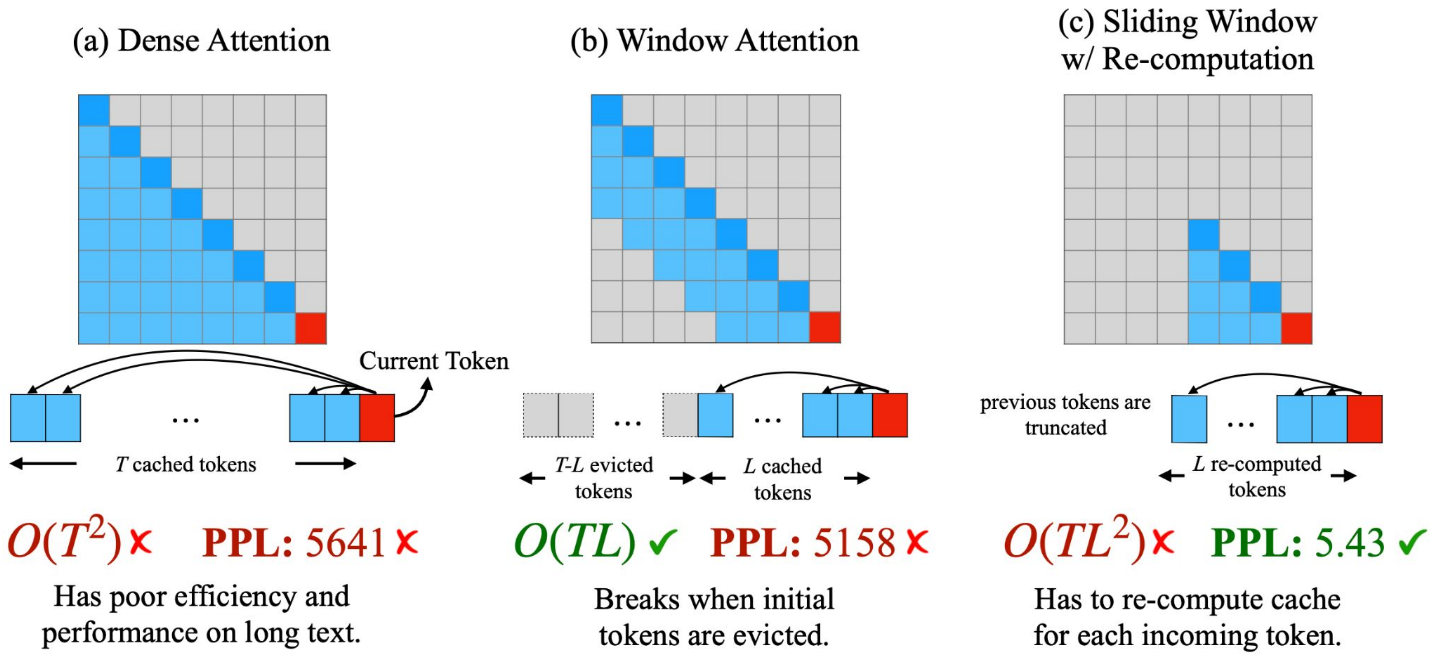
Improving Long Text Utilization

Model / Algorithm

System

StreamLLM and its baselines

Length extrapolation + System



Carnegie Mellon University

https://github.com/mit-han-lab/streaming-llm/blob/main/assets/StreamingLLM.pdf

Problems with dense and window attention

Perplexity

Dense and window attention fails when we generate a significant amount of tokens, especially when the text length is greater than cache size.

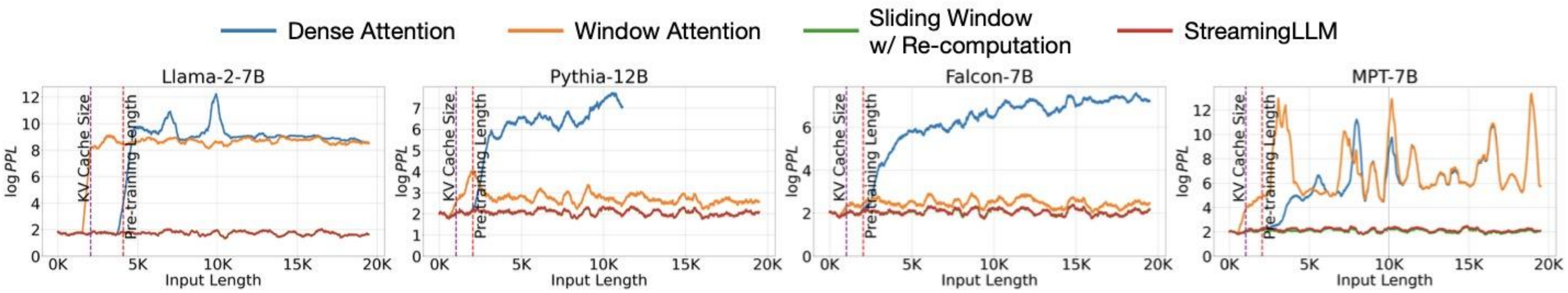


Figure 3: Language modeling perplexity on texts with 20K tokens across various LLM. Observations reveal consistent trends: (1) Dense attention fails once the input length surpasses the pre-training attention window size. (2) Window attention collapses once the input length exceeds the cache size, i.e., the initial tokens are evicted. (3) StreamingLLM demonstrates stable performance, with its perplexity nearly matching that of the sliding window with re-computation baseline.

Problems with window attention

Removal of first tokens

Window attention follows the sliding window algorithm, and it removes the consideration for the initial tokens when it spikes the cache.

But…

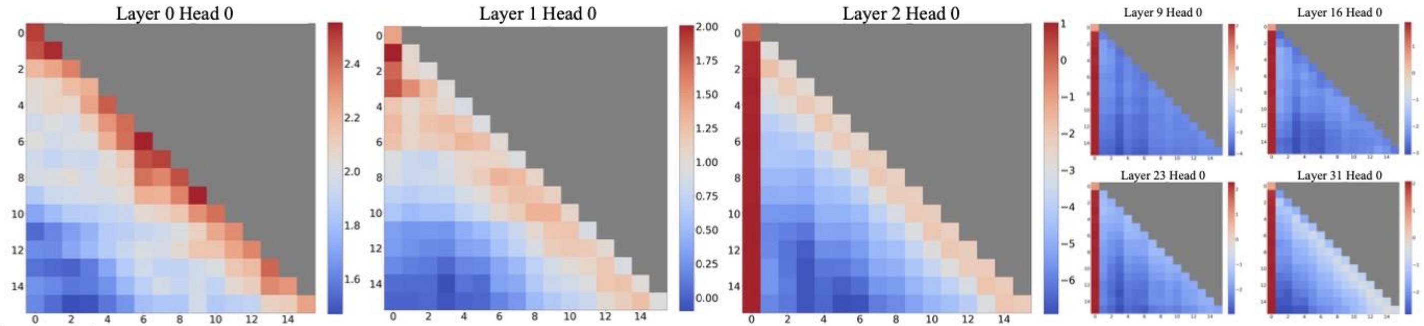


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two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

Attention sinks:

The initial tokens are important!

$$\text{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

Experiment results:

Why initial tokens?

Initial tokens are visible to all subsequent tokens later tokens are only visible to a limited set of subsequent tokens.

Therefore, initial tokens are easier to be trained

Table 1: Window attention has poor performance on long text. The perplexity is restored when we reintroduce the initial four tokens alongside the recent 1020 tokens (4+1020). Substituting the original four initial tokens with linebreak tokens "\n" (4"\n"+1020) achieves comparable perplexity restoration. Cache config x+y denotes adding x initial tokens with y recent tokens. Perplexities are measured on the first book (65K tokens) in the PG19 test set.

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

Carnegie Mellon University

StreamingLLM

StreamingLLM focuses on positions **within the cache** rather than **those in the original text** when determining the relative distance and adding positional information to tokens

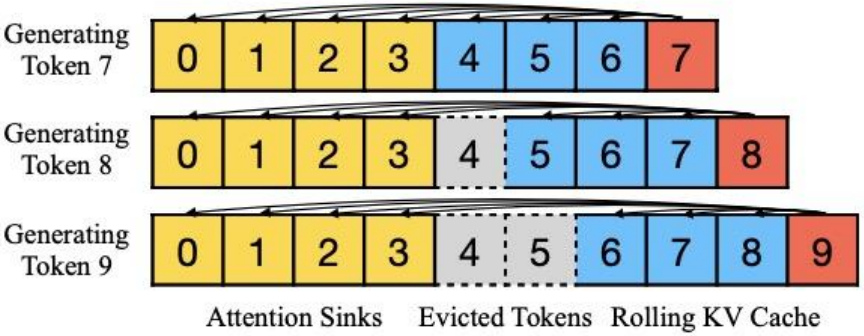


Figure 4: The KV cache of StreamingLLM.

Softmax Off-by-One

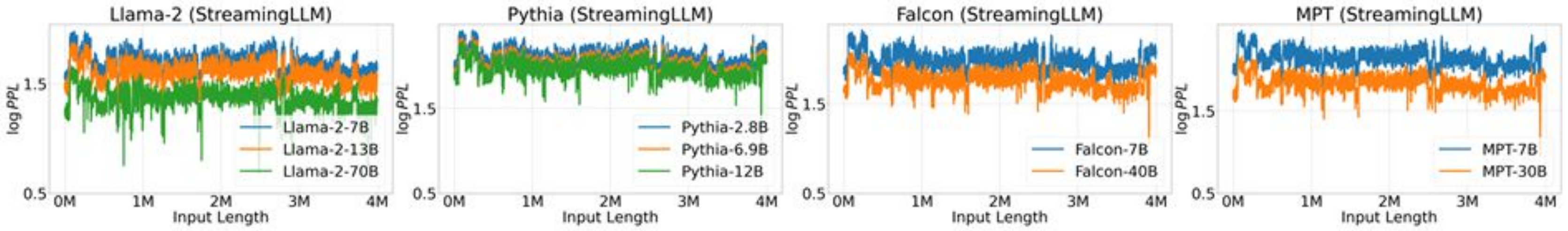
Softmax function that takes KV cache and intentional inclusion of attention sink into consideration

() Equivalent to using a token with K = V = 0s in attention ⇒ ZeroSink

$$\text{SoftMax}_1(x)_i = \frac{e^{x_i}}{1 + \sum_{j=1}^{N} e^{x_j}}, \qquad (2)$$

Performance on Long Context LLMs

- StreamingLLM can handle up to 4 million tokens across different models
- Perplexity remains stable
- Test set: PG19 (100 long books)



[Attention Sinks, Xiao et al. 2023]

Carnegie Mellon University

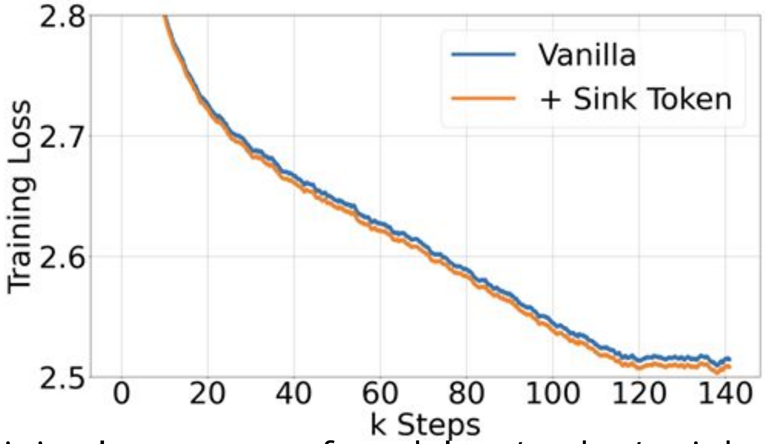
Experiments : Pre-Training with a Sink Token

- Model: Pythia-160M LM
  - Vanilla
    - *Original training setting*
  - With Sink Token
    - *Introduce a sink token at the start of every training sample*
- Devices: 8xA6000 NVIDIA GPUs
- Dateset: deduplicated Pile Dataset
- Configs: Pythia training configurations with batch size reduced to 256
- Trained for 143,000 steps

[Attention Sinks, Xiao et al. 2023] [Pythia-160M, Bider man et al. 2023] [Pile Dataset, Gao et al. 2020]

Results on Pre-Training with a Sink Token

- Convergence
  - No negative impact on model convergence



Pre-training loss curves of models w/and w/o sink tokens.

[Attention Sinks, Xiao et al. 2023]

Carnegie Mellon University

Results on Pre-Training with a Sink Token

- Normal model performance
  - Evaluate on 7 NLP benchmarks
  - Model with sink token performs similar to vanilla approach

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 Sinks, Xiao et al. 2023]

Results on Pre-Training with a Sink Token

- Streaming Performance
  - Vanilla model requires additional multiple tokens to enable stable perplexity
  - Zero Sink token shows improvement, but still needs other initial tokens
  - Model trained with a Sink Token achieves stable perplexity with only the sink token

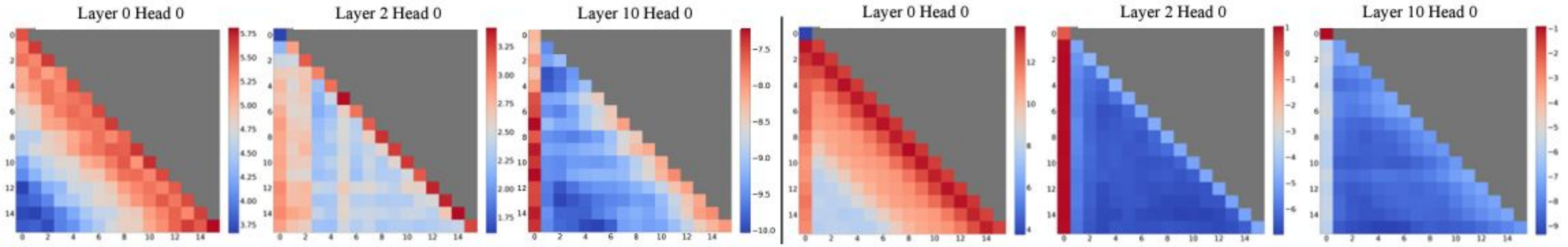
Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02

Figure: Comparison of vanilla attention with prepending a zero token and a learnable sink token during pretraining.

[Attention Sinks, Xiao et al. 2023]

Results on Pre-Training with a Sink Token

- Attention Visualization
    - Without attention sink token
        - *Local attention in lower layer*
        - *Increased attention to initial token in deeper layers*
    - With attention sink token
        - *Strong attention to sink*
        - *Reduced attention to other initial tokens*



Pre-Trained without Sink Token   Pre-Trained with Sink Token

[Attention Sinks, Xiao et al. 2023]

Results on Streaming QA

- Multi-round question-answering: concatenate ARC dataset
- Dense attention: results in OOM
- Window attention: poor accuracy
- StreamingLLM: high accuracy match one-shot sample-to-sample performance

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

[Attention Sinks, Xiao et al. 2023]

Carnegie Mellon University

Results on Streaming QA

- Creation of long context QA dataset suitable for StreamingLLM (StreamEval)
- Inspired by LongEval
- Query the model every 10 lines of the new information
- Answer from previous 20 lines

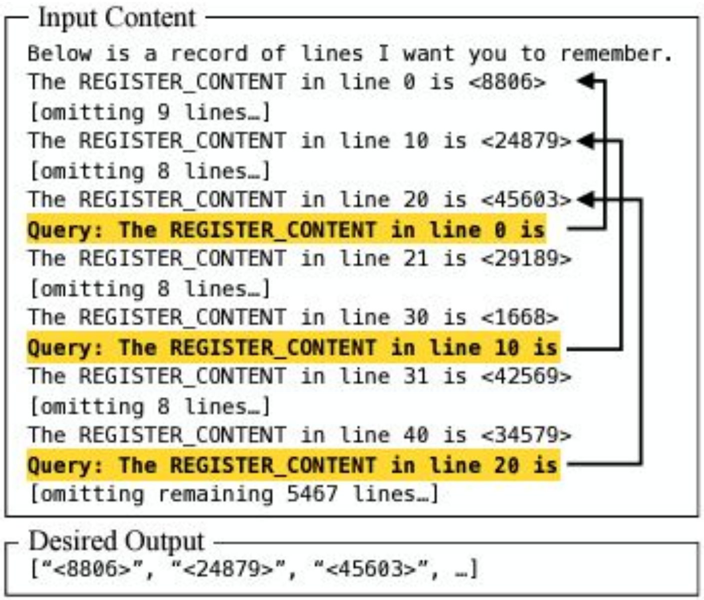


Figure 8: The first sample in StreamEval.

[Attention Sinks, Xiao et al. 2023]

Carnegie Mellon University

Results on Streaming QA

- Dense attention and window attention fail as context gets longer
- LLMs using Streaming LLM maintain accuracy with long input upto 120k tokens

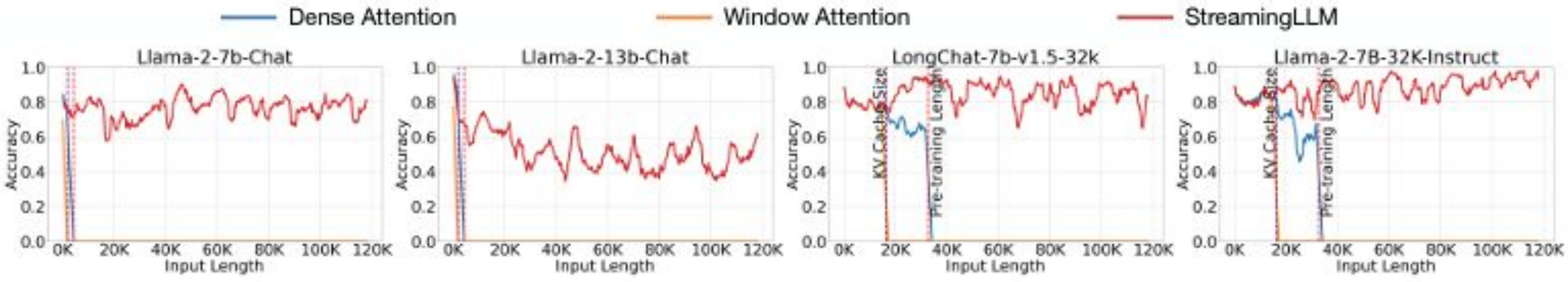


Figure 9: Performance on the StreamEval benchmark. Accuracies are averaged over 100 samples.

[Attention Sinks, Xiao et al. 2023]

Carnegie Mellon University

Ablation Study

- Number of attention sinks needed to recover perplexity
  - 4 attention sinks

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

Carnegie Mellon University

Effects of Cache Size

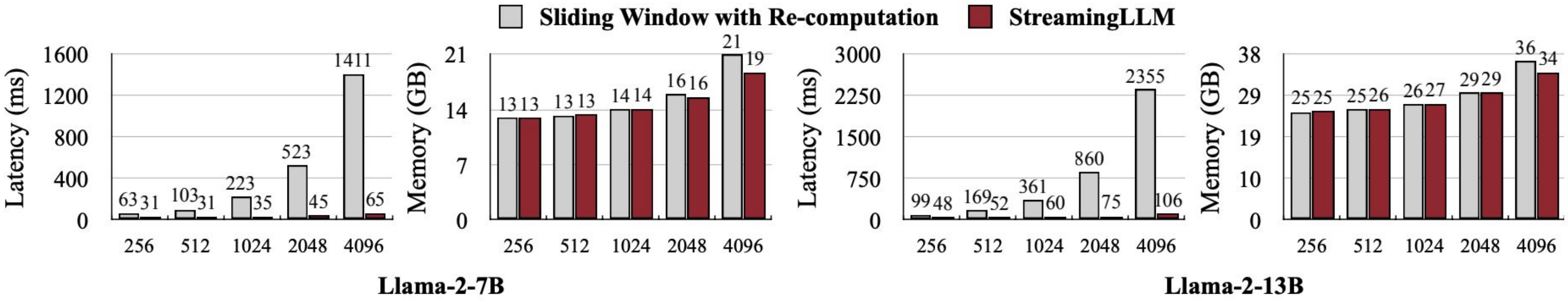
- Increasing the cache size in doesn't consistently yield a decrease in perplexity
  - Models may not fully utilize the provided context

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
Llama-2-7B	9.73	9.32	**9.08**	9.59

Efficiency

- Comparison baseline:
  - Sliding window with recomputation
    - *Computationally heavy because of quadratic attention computation within its window*
  - StreamingLLM
    - *Speedup 22.2x over the baseline making LLMs feasible for real-time streaming*



Carnegie Mellon University

Accuracy on StreamEval with Increasing Query-answer Distance

- Cannot extend the context length of language models
- Inability to fully utilize context information

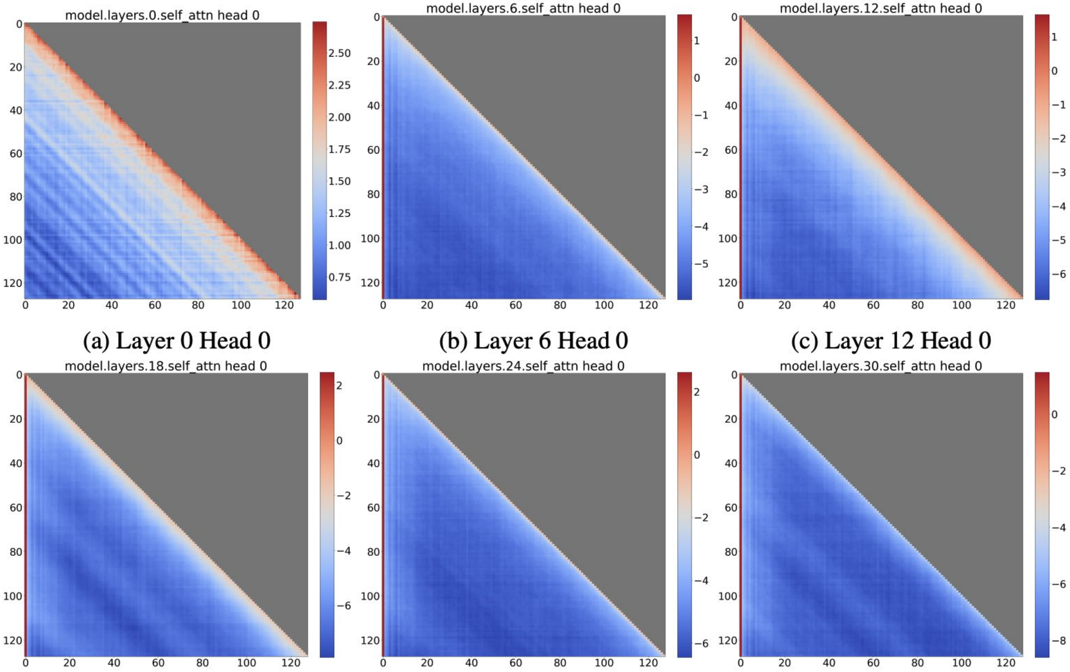
Llama-2-7B-32K-Instruct		Cache Config			
Line Distances	Token Distances	4+2044	4+4092	4+8188	4+16380
20	460	85.80	84.60	81.15	77.65
40	920	80.35	83.80	81.25	77.50
60	1380	79.15	82.80	81.50	78.50
80	1840	75.30	77.15	76.40	73.80
100	2300	0.00	61.60	50.10	40.50
150	3450	0.00	68.20	58.30	38.45
200	4600	0.00	0.00	62.75	46.90
400	9200	0.00	0.00	0.00	45.70
600	13800	0.00	0.00	0.00	28.50
800	18400	0.00	0.00	0.00	0.00
1000	23000	0.00	0.00	0.00	0.00

Comparison of StreamingLLM against the Default Truncation baseline in LongBench

Llama2-7B-chat	Single-Document QA		Multi-Document QA		Summarization	
	NarrativeQA	Qasper	HotpotQA	2WikiMQA	GovReport	MultiNews
Truncation 1750+1750	18.7	19.2	25.4	32.8	27.3	25.8
StreamingLLM 4+3496	11.6	16.9	21.6	28.2	23.9	25.5
StreamingLLM 1750+1750	18.2	19.7	24.9	32.0	26.3	25.9

Carnegie Mellon University

Llama-2-7B Attention on Longer Sequences

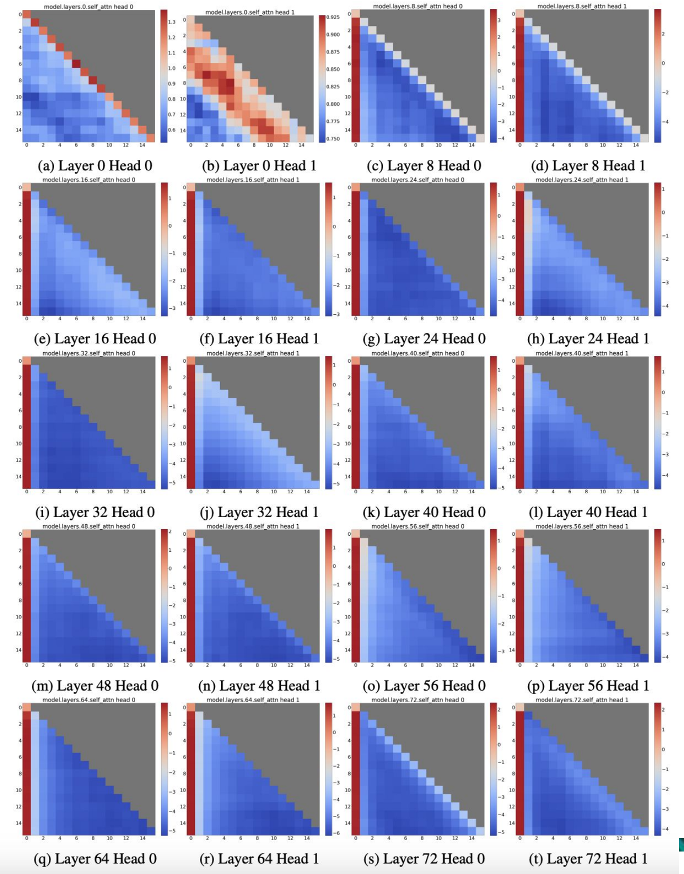


(a) Layer 0 Head 0 (b) Layer 6 Head 0 (c) Layer 12 Head 0

(d) Layer 18 Head 0 (e) Layer 24 Head 0 (f) Layer 30 Head 0

Carnegie Mellon University

Average Attention Logits in Llama-2-70B over 256 Sentences



(a) Layer 0 Head 0 (b) Layer 0 Head 1 (c) Layer 8 Head 0 (d) Layer 8 Head 1

(e) Layer 16 Head 0 (f) Layer 16 Head 1 (g) Layer 24 Head 0 (h) Layer 24 Head 1

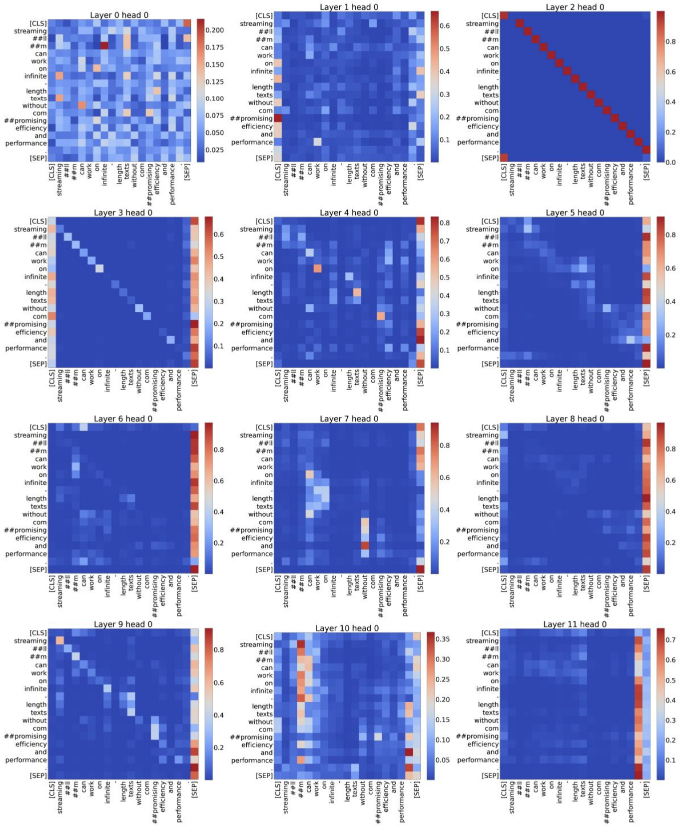
(i) Layer 32 Head 0 (j) Layer 32 Head 1 (k) Layer 40 Head 0 (l) Layer 40 Head 1

(m) Layer 48 Head 0 (n) Layer 48 Head 1 (o) Layer 56 Head 0 (p) Layer 56 Head 1

(q) Layer 64 Head 0 (r) Layer 64 Head 1 (s) Layer 72 Head 0 (t) Layer 72 Head 1

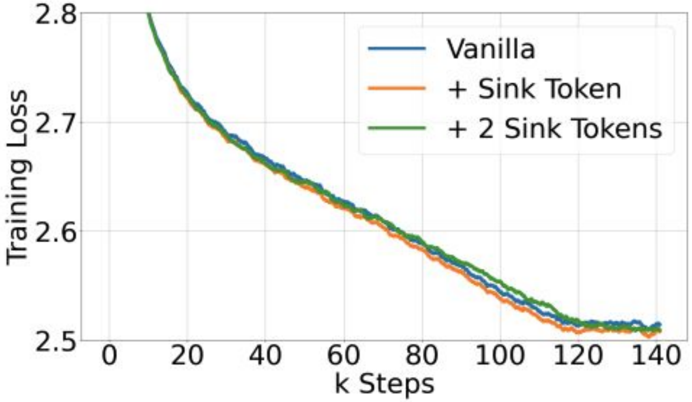
Carnegie Mellon University

Attention Maps in BERT-base-uncased



Carnegie Mellon University

Using more Sink Tokens in Pre-training Stage



Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
+ 1 Sink Token	1235	**18.01**	18.01	18.02
+ 2 Sink Tokens	1262	25.73	18.05	18.05

Carnegie Mellon University

Zero-shot Accuracy across 7 NLP Benchmarks

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
+ 1 Sink Token	**19.6**	**45.6**	**29.8**	**39.9**	**16.6**	62.6	**50.8**
+ 2 Sink Tokens	18.7	45.6	29.6	37.5	15.8	**64.3**	50.4

Conclusion and Future Works

- Conclusion
  - StreamingLLM
    - *Handles unlimited text lengths without fine-tuning*
    - *Utilizes "attention sinks" with recent tokens for enhanced efficiency*
    - *Can model texts up to 4 million tokens*
  - Advancements & Benefits
    - *Pre-training with dedicated sink token improves streaming performance*
    - *Facilitates the streaming deployment of LLMs*
- Future work
  - Enhancing LLM models' capabilities to utilize extensive contexts better