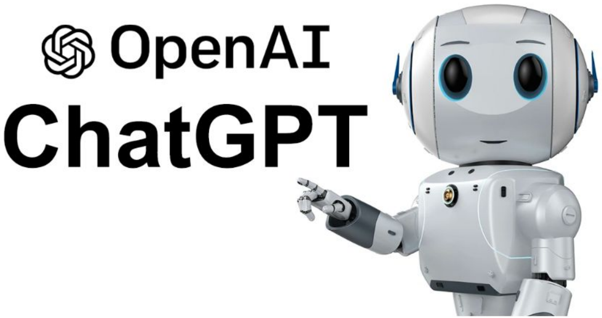
Carnegie Mellon University

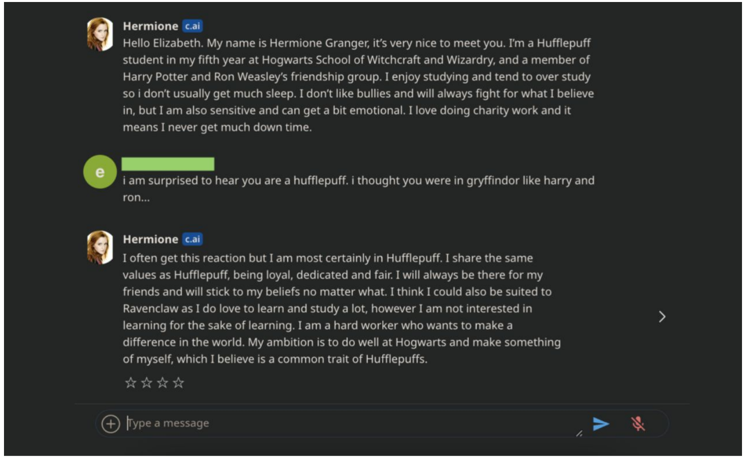
Efficient Streaming LMs with Attention Sinks

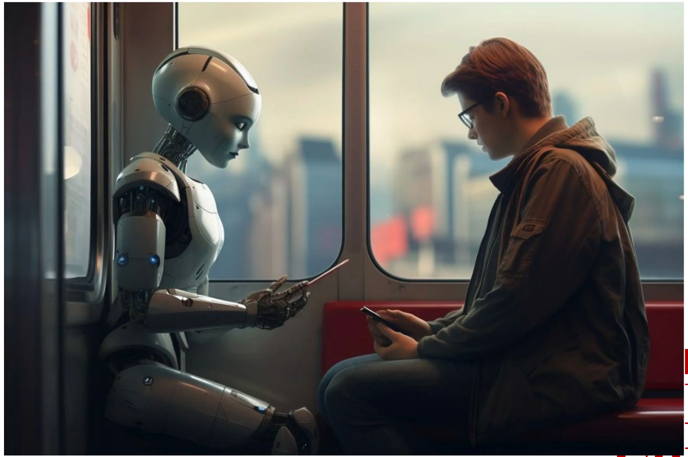
Yitong Chen, Ruihang Lai, Judy Jin, Nasrin Kalanat

11868 LLM System

Motivation: Use cases



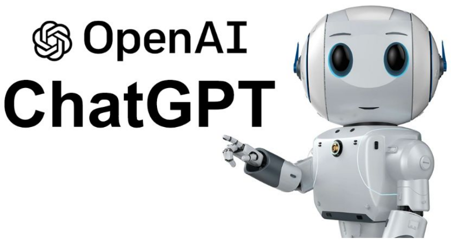




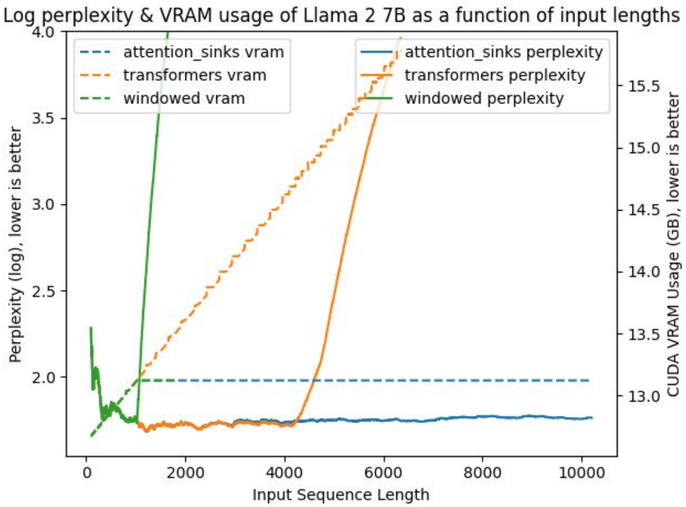
https://github.com/mit-han-lab/streaming-llm/blob/main/assets/StreamingLLM.pdf

Challenges of Deploying LLMs in Streaming Applications

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



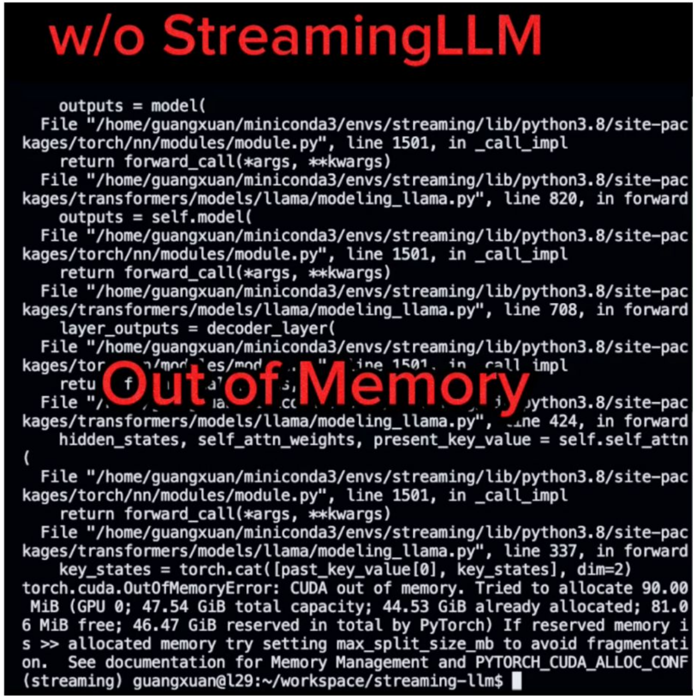
- Challenges
  - Extensive memory consumption during the decoding stage.
  - Inability of popular LLMs to generalize to longer text sequences.



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

Carnegie Mellon University

Directions of Approaching the Problem?

Making more use of your context

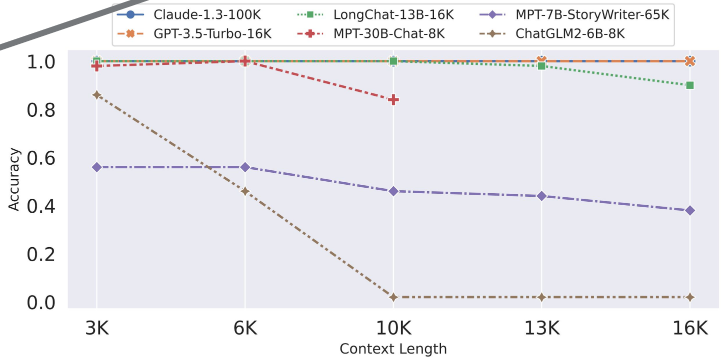
Length extrapolation

Context Window Extension

Improving Long Text Utilization

Model / Algorithm

System



Carnegie Mellon University

Directions of Approaching the Problem?

Applying LLMs with fixed context size on longer content

Length extrapolation

Context Window Extension

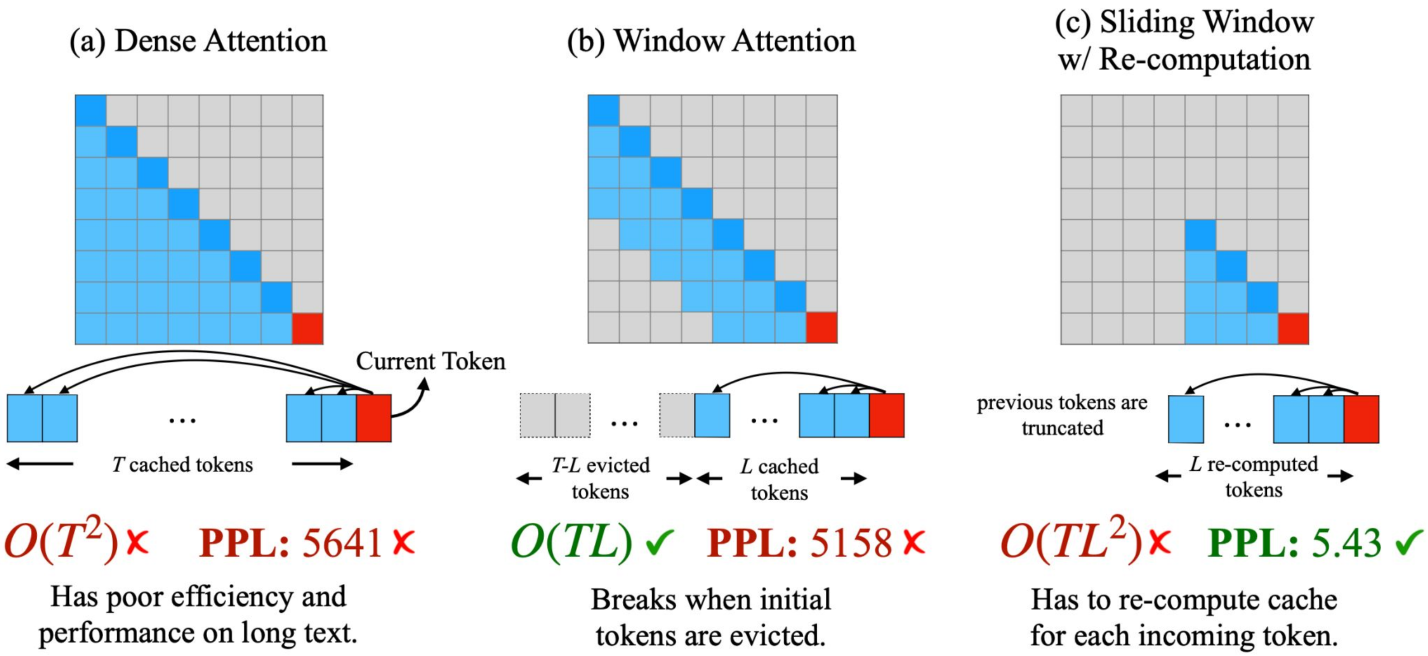
Improving Long Text Utilization

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.

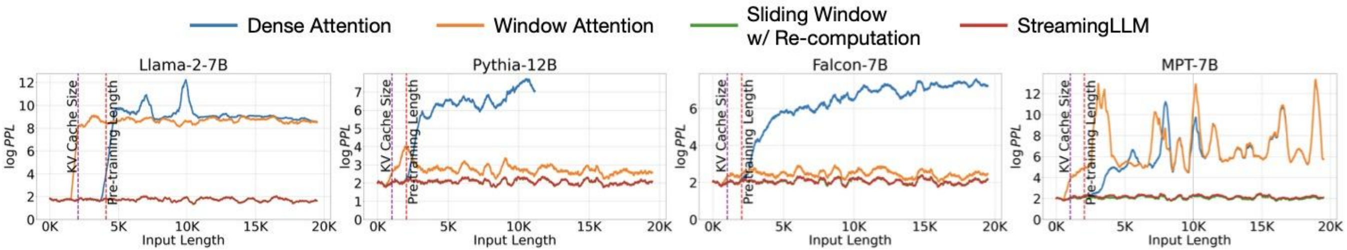


Figure 3: Language modeling perplexity on texts with 20K tokens across various LLM. Observations reveal consistent trends: (1) Dense attention fails once the input length surpasses the pre-training attention window size. (2) Window attention collapses once the input length exceeds the cache size, i.e., 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…

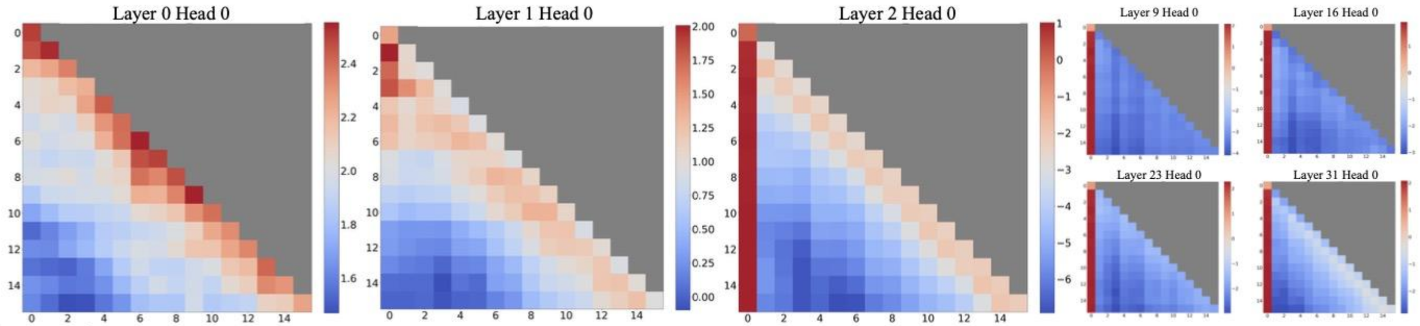


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two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

Attention sinks:

The initial tokens are important!

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

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

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

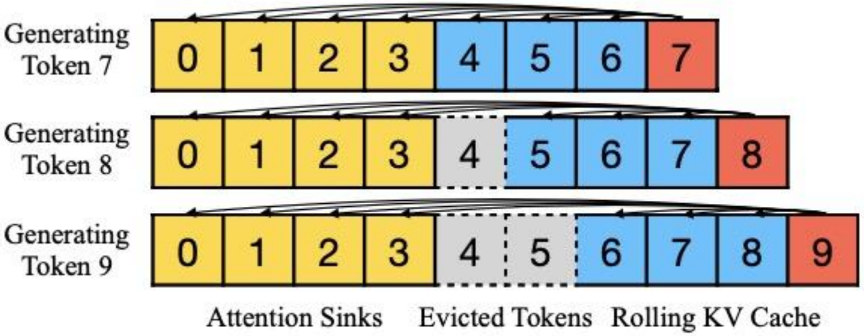
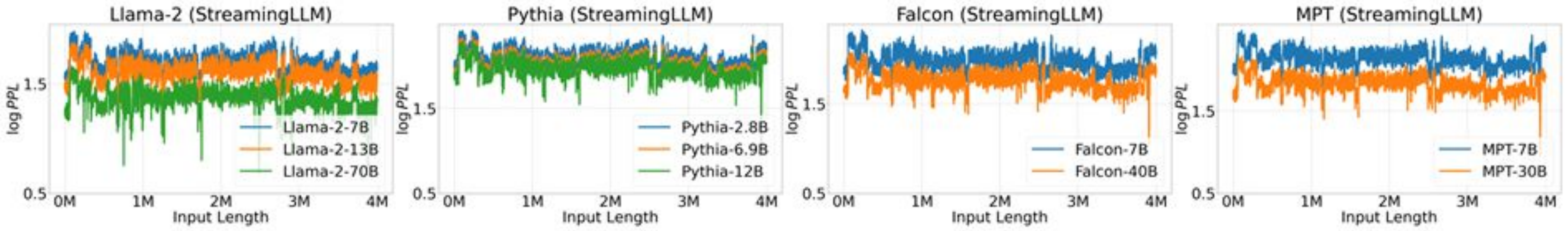


Figure 4: The KV cache of StreamingLLM.

Carnegie Mellon University

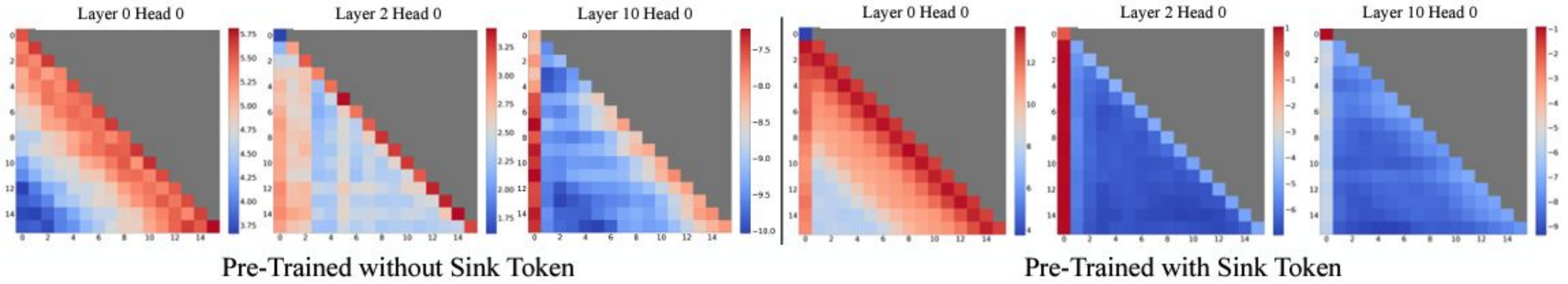
Performance on Long Context LLMs

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



Results on Pre-Training with a Sink Token

- 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



Carnegie Mellon University

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

Carnegie Mellon University

Results on Streaming QA

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

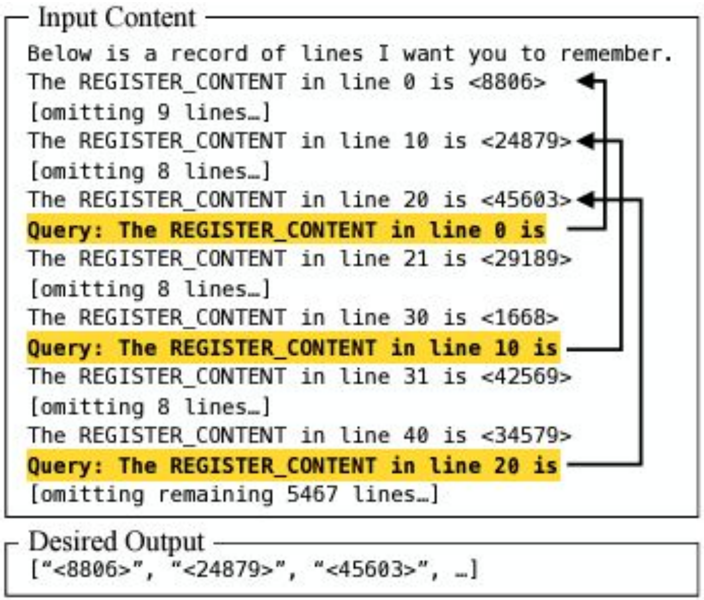


Figure 8: The first sample in StreamEval.

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

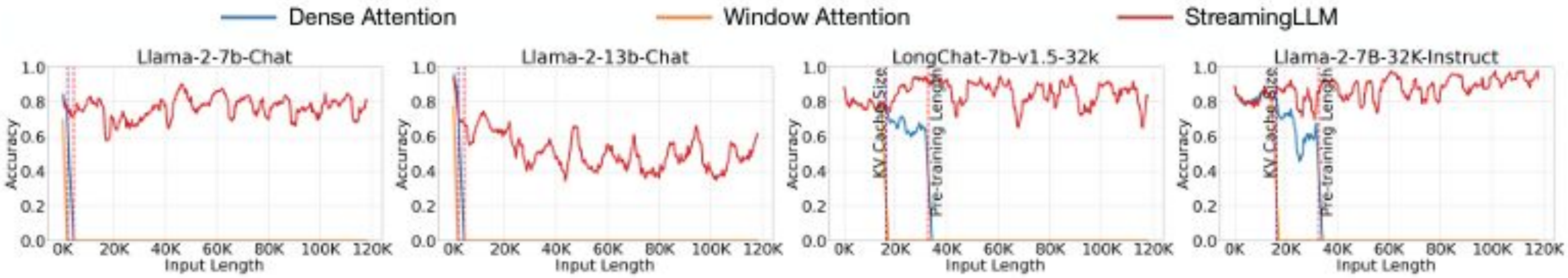


Figure 9: Performance on the StreamEval benchmark. Accuracies are averaged over 100 samples.

Ablation Study

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

Table 2: Effects of reintroduced initial token numbers on StreamingLLM. (1) Window attention (0+y) has a drastic increase in perplexity. (2) Introducing one or two initial tokens usually doesn't suffice to fully restore model perplexity, indicating that the model doesn't solely use the first token as the attention sink. (3) Introducing four initial tokens generally suffices; further additions have diminishing returns. Cache config x+y denotes adding x initial tokens to y recent tokens. Perplexities are evaluated on 400K tokens in the concatenated PG19 test set.

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

Pre-training with a Dedicated Attention Sink Token

- Can we train a LLM that need only one single attention sink? Yes!
- Solution
  - an extra learnable token at the beginning of all training samples to act as a dedicated attention sink
    - *retains performance in streaming cases with just this single sink token*
    - *contrasting with vanilla models that require multiple initial tokens*

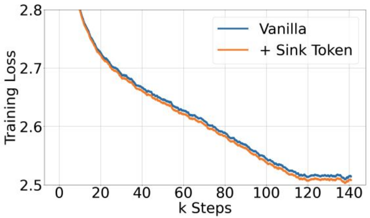


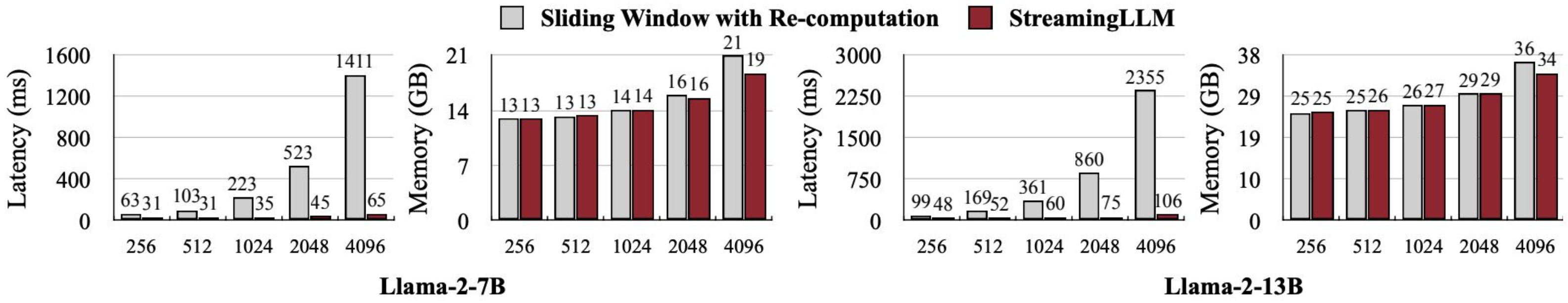
Figure 6: Pre-training loss curves of models w/ and w/o sink tokens. Two models have a similar convergence trend.

Table 3: Comparison of vanilla attention with prepending a zero token and a learnable sink token during pre-training. To ensure stable streaming perplexity, the vanilla model required several initial tokens. While Zero Sink demonstrated a slight improvement, it still needed other initial tokens. Conversely, the model trained with a learnable Sink Token showed stable streaming perplexity with only the sink token added. Cache config x+y denotes adding x initial tokens with y recent tokens. Perplexity is evaluated on the first sample in the PG19 test set.

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

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*



Conclusion and Future Works

- Conclusion
  - StreamingLLM: A Novel Framework
    - *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*
    - *Decouples pre-training window size from text generation length*
    - *Facilitates the streaming deployment of LLMs*
- Future work
  - Enhancing LLM models’ capabilities to utilize extensive contexts better

Carnegie Mellon University