

Efficient Streaming Language Models with Attention Sinks

Guangxuan Xiao¹, Yuandong Tian², Beidi Chen³, Song Han^{1,4}, Mike Lewis²

Massachusetts Institute of Technology¹ Meta Al² Carnegie Mellon University³ NVIDIA⁴

Motivation: Use cases





Hermione c.ai Hello Elizabeth. My name is Hermione Granger, it's very nice to meet you. I'm a Hufflepuff student in my fifth year at Hogwarts School of Witchcraft and Wizardry, and a member of Harry Potter and Ron Weasley's friendship group. I enjoy studying and tend to over study so i don't usually get much sleep. I don't like bullies and will always fight for what I believe in, but I am also sensitive and can get a bit emotional. I love doing charity work and it means I never get much down time.



i am surprised to hear you are a hufflepuff. i thought you were in gryffindor like harry and ron..



Hermione c.ai I often get this reaction but I am most certainly in Hufflepuff. I share the same values as Hufflepuff, being loyal, dedicated and fair. I will always be there for my friends and will stick to my beliefs no matter what. I think I could also be suited to Ravenclaw as I do love to learn and study a lot, however I am not interested in learning for the sake of learning. I am a hard worker who wants to make a difference in the world. My ambition is to do well at Hogwarts and make something of myself, which I believe is a common trait of Hufflepuffs.

 $\pounds \And \pounds & \pounds$







>

> 🕺

Challenges of Deploying LLMs in Streaming Applications

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



"Alexa, open Showcase Cinemas."

- Challenges:
 - Extensive memory consumption during lacksquarethe decoding stage.
 - Inability of popular LLMs to generalize to \bullet longer text sequences.



https://github.com/tomaarsen/attention_sinks





Challenges of Deploying LLMs in Streaming Applications w/ StreamingLLM w/o StreamingLLM

S=0 python examples/run_streaming_llama.py Loading model from lmsys/vicuna-13b-v1.3 ... Loading checkpoint shards: 67%

- (streaming) guangxuan@l29:~/workspace/streaming-llm\$ CUDA_VISIBLE_DEVICE (streaming) guangxuan@l29:~/workspace/streaming-llm\$ CUDA_VISIBLE_DEVICES=1 py thon examples/run_streaming_llama.py --enable_streaming
 - Loading model from lmsys/vicuna-13b-v1.3 ...
 - 2/3 [00:09<00:04, 4.94s/it] Loading checkpoint shards: 67%]

| 2/3 [00:09<00:04, 4.89s/it]</pre>







Challenges of Deploying LLMs in Streaming Applications

w/o StreamingLLM

| | "0000" | "0"0"000 | 0000" | | | | | | |
|------------------|----------|-----------|-----------|----------|---------|----------|--------|-----------|----|
| | 00000000 | aaaa"a"a' | 'a"aaaa"(| a"a"a""" | | aaaaaaaa | aaaaaa | | |
| 0000000 a'''' | 00000000 | 000000000 | 0000"0000 | 00000000 | 000"0"" | | | ······0-1 | t0 |
| | | | | | | | | | |

Model Performance ASSISTANT: 0000000-t-t-t-t"

USER: Write a C++ prog Break Sibonacci number using recursi on.

USER: Now we define a sequence of numbers in which each number is the su m of the three preceding ones. The first three numbers are 0, -1, -1. Wr ite a program to find the nth number.

ASSISTANT: 0-a-a-eah00000000000

USER: Write a simple website in HTML. When a user clicks the button, it shows a random joke from a list of 4 jokes.

w/o StreamingLLM outputs = model(File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/torch/nn/modules/module.py", line 1501, in _call_impl return forward_call(*args, **kwargs) File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/transformers/models/llama/modeling_llama.py", line 820, in forward outputs = self.model(File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/torch/nn/modules/module.py", line 1501, in _call_impl return forward_call(*args, **kwargs) File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/transformers/models/llama/modeling_llama.py", line 708, in forward layer_outputs = decoder_layer(File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/tor _______ es/mod _____ retu 🚺 f File "/ Jython3.8/site-pac kages/transformers/models/llama/modeling_llama.py", ____ne 424, in forward hidden_states, self_attn_weights, present_key_value = self.self_attn File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/torch/nn/modules/module.py", line 1501, in _call_impl return forward_call(*args, **kwargs) File "/home/guangxuan/miniconda3/envs/streaming/lib/python3.8/site-pac kages/transformers/models/llama/modeling_llama.py", line 337, in forward key_states = torch.cat([past_key_value[0], key_states], dim=2) torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 90.00 MiB (GPU 0; 47.54 GiB total capacity; 44.53 GiB already allocated; 81.0 6 MiB free; 46.47 GiB reserved in total by PyTorch) If reserved memory i s >> allocated memory try setting max_split_size_mb to avoid fragmentati on. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF (streaming) guangxuan@l29:~/workspace/streaming-llm\$



The Problem of Long Context: Large KV Cache The KV cache could be large with long context

- \bullet tokens so that we can perform the attention computation, namely the **KV cache**
 - Only need the **current** query token \bullet



$$a_{ij} = \frac{\exp(q_i^\top k_j / \sqrt{d})}{\sum_{t=1}^i \exp(q_i^\top k_t / \sqrt{d})}, \ o_i = \sum_{j=1}^i a_{ij} v_j$$

Image credit: https://awsdocs-neuron.readthedocs-hosted.com/en/latest/general/appnotes/transformers-neuronx/generative-llm-inference-with-neuron.html

During Transformer decoding (GPT-style), we need to store the Keys and Values of all previous

The Problem of Long Context: Large KV Cache

The KV cache could be large with long context

- We can calculate the memory required to store the KV cache \bullet
- Take Llama-2-7B as an example \bullet

layers kv-heads batchsize length n_{emd} Now we calculate the KV cache size under BS = 4 and different sequence lengths.

Quickly larger than model weights lacksquare

BS

* $\underline{32}$ * $\underline{32}$ * $\underline{128}$ * \underline{N} * $\underline{2}$ * 2bytes = $0.5MB \times BS \times N$ K&V

*FP*16

The Limits of Window Attention

- A natural approach window attention: caching only the most recent Key-Value states. lacksquare
- is evicted. (b) Window Attention

Drawback: model collapses when the text length surpasses the cache size, when the initial token

Difficulties of Other Methods

the sequence length; <u>perplexity</u> <u>explodes</u> after exceeding the max context length.

perplexity explodes after sequence length exceeds the KV cache size (first token evicted).

KV cache size is <u>constant</u>.

The "Attention Sink" Phenomenon

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.

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

Observation: initial tokens have large attention scores, even if they're not semantically significant. **Attention Sink:** Tokens that disproportionately attract attention irrespective of their relevance.

Understanding Why Attention Sinks Exist The Rationale Behind Attention Sinks

 SoftMax operation's role in creating attention Does the importance of the initial tokens arise sinks — attention scores have to sum up to one from their **position** or their **semantics**? for all contextual tokens.

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

Initial tokens' advantage in becoming sinks due lacksquareto their visibility to subsequent tokens, rooted in autoregressive language modeling.

 $P(S) = P(Where) \times P(are | Where) \times P(we | Where are) \times P(going | Where are we)$

- We found adding initial four "\n"s can also recover perplexity.
- Therefore, it is **position**!

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

StreamingLLM: Using Attention Sinks for Infinite Streams

- without additional training.
- stabilize the model's behavior.

(d) StreamingLLM (ours)

Can perform efficient and stable language modeling on long texts.

Objective: Enable LLMs trained with a finite attention window to handle infinite text lengths

Key Idea: preserve the KV of attention sink tokens, along with the sliding window's KV to

Positional Encoding Assignment

Use positions in the cache instead of those in the original text.

Streaming Performance

- Dense attention fails beyond pre-training attention window size.
- Window attention fails after input exceeds cache size (initial tokens evicted).
- \bullet

Comparison between dense attention, window attention, and sliding window w/ re-computation.

StreamingLLM shows stable performance; perplexity close to sliding window with re-computation baseline.

Streaming Performance Super Long Language Modeling

lacksquaremodel up to 4 million tokens.

With StreamingLLM, model families include Llama-2, MPT, Falcon, and Pythia can now effectively

Efficiency

- **Comparison baseline:** The sliding window with re-computation, a method that is computationally heavy due to quadratic attention computation within its window.
- StreamingLLM provides up to 22.2x speedup over the baseline, making LLMs for real-time streaming applications feasible.

Llama-2-7B

Llama-2-13B

Ablation Study: #Attention Sinks

- The number of attention sinks that need to be introduced to recover perplexity. lacksquare
 - 4 attention sinks are generally enough.

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

- \bullet
- dedicated attention sink.
- \bullet token, contrasting with vanilla models that require multiple initial tokens.

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

Idea: Why 4 attention sinks? Can we train a LLM that need only one single attention sink? Yes! **Method:** Introduce an extra learnable token at the start of all training samples to act as a

Result: This pre-trained model retains performance in streaming cases with just this single sink

| I | | |
|---|--|--|
| | | |
| | | |
| 1 | | |
| - | | |
| | | |
| - | | |
| | | |
| | | |

| Cache Config | 0+1024 | 1+1023 | 2+1022 | 4+1020 |
|----------------------|----------------|------------------------------|--------|--------|
| Vanilla Zero Sink | 27.87 20214 | 18.49 | 18.05 | 18.05 |
| Learnable Sink | 1235 | 19.90 18.01 | 18.01 | 18.02 |

Attention Sinks in Other Transformers Encoder Models: ViT and BERT

Input

DeiT-III

Without registers DINOv2

CLIP

ViT

Vision Transformers Need Registers

Can StreamingLLM give us infinite context?

- Non-stop chatting \neq Infinite context lacksquare
- Tokens that are evicted from cache cannot be attended.

| ama-2-7B | -32K-Instruct | Cache Config | | | | | |
|-----------|-----------------|--------------|--------|--------|--------|--|--|
| Distances | Token Distances | 4+2044 | 4+4092 | 4+8188 | 4+1638 | | |
| | 460 | 85.80 | 84.60 | 81.15 | 77.65 | | |
| | 920 | 80.35 | 83.80 | 81.25 | 77.50 | | |
| | 1380 | 79.15 | 82.80 | 81.50 | 78.50 | | |
| | 1840 | 75.30 | 77.15 | 76.40 | 73.80 | | |
| | 2300 | 0.00 | 61.60 | 50.10 | 40.50 | | |
| | 3450 | 0.00 | 68.20 | 58.30 | 38.45 | | |
| | 4600 | 0.00 | 0.00 | 62.75 | 46.90 | | |
| | 9200 | 0.00 | 0.00 | 0.00 | 45.70 | | |
| | 13800 | 0.00 | 0.00 | 0.00 | 28.50 | | |
| | 18400 | 0.00 | 0.00 | 0.00 | 0.00 | | |
| | 23000 | 0.00 | 0.00 | 0.00 | 0.00 | | |

Thanks for Listening!

- We propose StreamingLLM, enabling the streaming deployment of LLMs. lacksquare
- Paper: <u>https://arxiv.org/abs/2309.17453</u>
- Code: <u>https://github.com/mit-han-lab/streaming-llm</u> 6.2K Stars
- Demo: <u>https://youtu.be/UgDcZ3rvRPg</u> ${\color{black}\bullet}$

| | w/o | Stre | eam | ning | LLM | |
|-------------|---------------------------|---|--------------------------------------|-------------------------|---------------|--------------|
| (str S=0 | reaming) gu python exa | uangxuan@l29 amples/run_s from lmsvs/ | :~/workspa treaming_ vicuna-13 | ace/stream: llama.py | ing-llm\$ CUI | DA_VISIBLE_D |
| Load | ling check | point shards | : 67% | | 2/3 [00:09< | 00:04, 4.94 |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

