SDML
ML Paper Review

December 2023

Efficient Streaming Language Models with Attention Sinks

https://arxiv.org/abs/2309.17453

Efficient Streaming Language Models with Attention Sinks

Guangxuan Xiao[1*] Yuandong Tian[2] Beidi Chen[3] Song Han[1] Mike Lewis[2]

[1] Massachusetts Institute of Technology
[2] Meta AI
[3] Carnegie Mellon University
https://github.com/mit-han-lab/streaming-llm

Abstract

Deploying Large Language Models (LLMs) in streaming applications such as multi-round dialogue, where long interactions are expected, is urgently needed but poses two major challenges. Firstly, during the decoding stage, caching previous tokens' Key and Value states (KV) consumes extensive memory. Secondly, popular LLMs cannot generalize to longer texts than the training sequence length. Window attention, where only the most recent KVs are cached, is a natural approach — but we show that it fails when the text length surpasses the cache size. We observe an interesting phenomenon, namely *attention sink*, that keeping the KV of initial tokens will largely recover the performance of window attention. In this paper, we first demonstrate that the emergence of *attention sink* is due to the strong attention scores towards initial tokens as a "sink" even if they are not semantically important. Based on the above analysis, we introduce StreamingLLM, an efficient framework that enables LLMs trained with a *finite length* attention window to generalize to *infinite sequence length* without any fine-tuning. We show that StreamingLLM can enable Llama-2, MPT, Falcon, and Pythia to perform stable and efficient language modeling with up to 4 million tokens and more. In addition, we discover that adding a placeholder token as a dedicated attention sink during pre-training can further improve streaming deployment. In streaming settings, StreamingLLM outperforms the sliding window recomputation baseline by up to 22.2× speedup. Code and datasets are provided in the link.

1 Introduction

Large Language Models (LLMs) (Radford et al., 2018; Brown et al., 2020; Zhang et al., 2022; OpenAI, 2023; Touvron et al., 2023a;b) are becoming ubiquitous, powering many natural language processing applications such as dialog systems (Schulman et al., 2022; Taori et al., 2023; Chiang et al., 2023), document summarization (Goyal & Durrett, 2020; Zhang et al., 2023a), code completion (Chen et al., 2021; Rozière et al., 2023) and question answering (Kamalloo et al., 2023). To unleash the full potential of pretrained LLMs, they should be able to efficiently and accurately perform long sequence generation. For example, an ideal ChatBot assistant can stably work over the content of recent day-long conversations. However, it is very challenging for LLM to generalize to longer sequence lengths than they have been pretrained on, e.g., 4K for Llama-2 Touvron et al. (2023b).

The reason is that LLMs are constrained by the attention window during pre-training. Despite substantial efforts to expand this window size (Chen et al., 2023; kaiokendev, 2023; Peng et al., 2023) and improve training (Dao et al., 2022; Dao, 2023) and inference (Pope et al., 2022; Xiao et al., 2023; Anagnostidis et al., 2023; Zhang et al., 2023b) efficiency for lengthy inputs, the acceptable sequence length remains intrinsically *finite*, which doesn't allow persistent deployments.

In this paper, we first introduce the concept of LLM streaming applications and ask the question:

Can we deploy an LLM for infinite-length inputs without sacrificing efficiency and performance?

[*] Part of the work done during an internship at Meta AI.

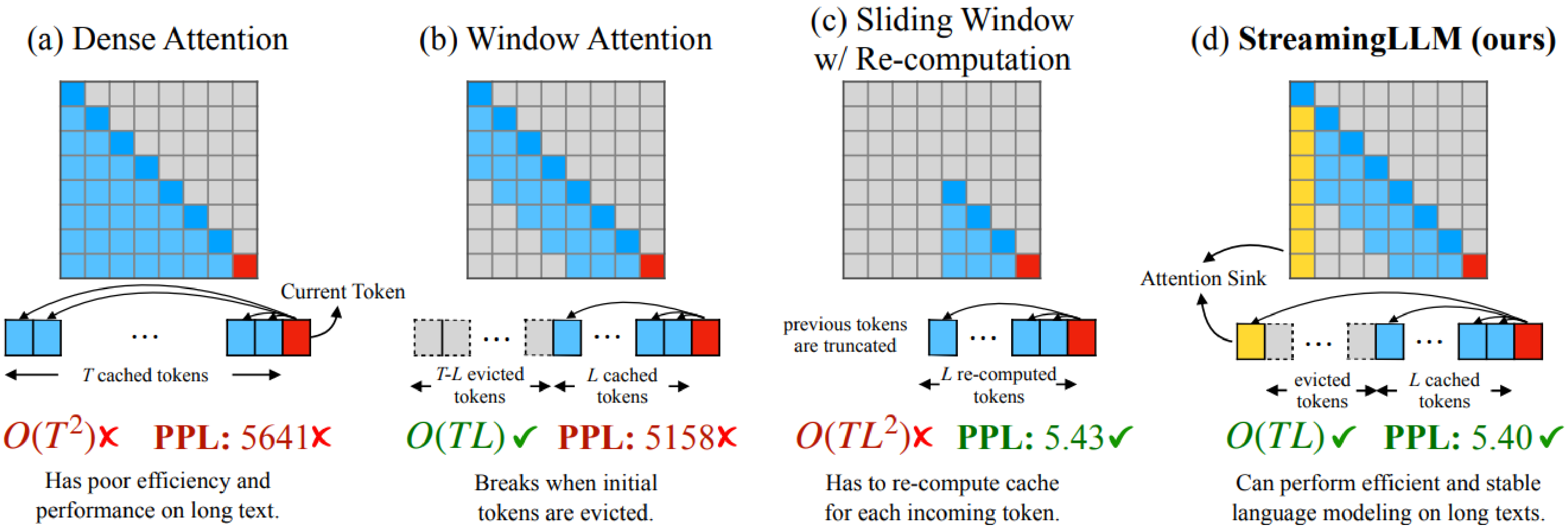
Attention sinks overview

- Efficient Streaming Language Models with Attention Sinks
- Guangxuan Xiao et al. from MIT

- Challenge: continue generate text, such as in a multi-round chatbot, even after the history has exceeded your LLM’s max context length
- The obvious solution is to use a sliding window of most recent tokens, but this leads to severe performance degradation
- Prior “Heavy Hitter” paper showed some tokens get a lot of attention. But that paper didn’t identify which ones.
- Keeping an attention sink of first token(s) solves performance problem

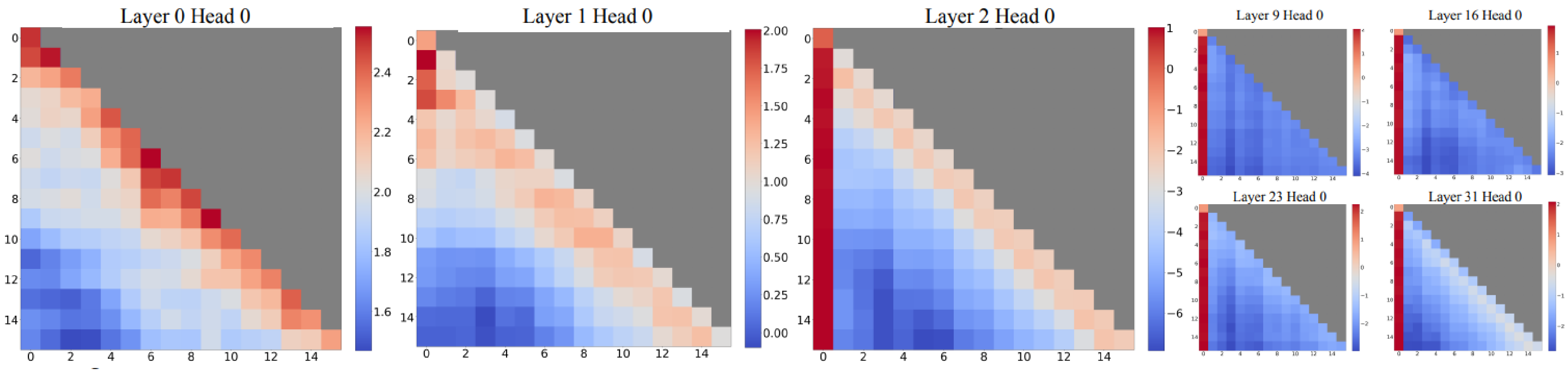
Long context approaches

- Regular sliding window with cached KV has bad quality (perplexity)
- Re-computation with sliding window doesn't cache, so hurts speed
- Attention sink approach uses caching and maintains text quality



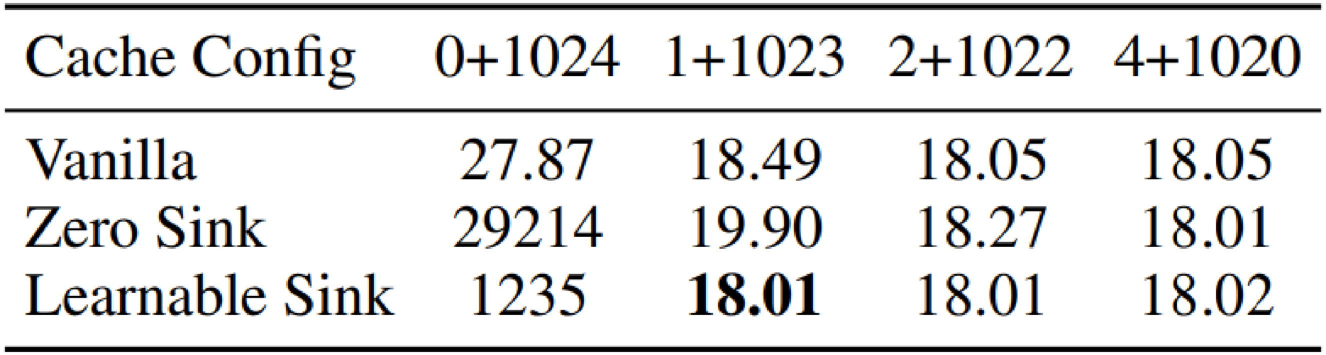
Why sliding window performance is impacted

- The attention maps in autoregressive LLMs put a lot of attention on the first token(s), which they name attention sinks
  - This was seen in the transformer circuits work by Anthropic https://transformer-circuits.pub/2021/framework/index.html
  - Test showed it is the position, not the content of first tokens, that matters



Attention sinks

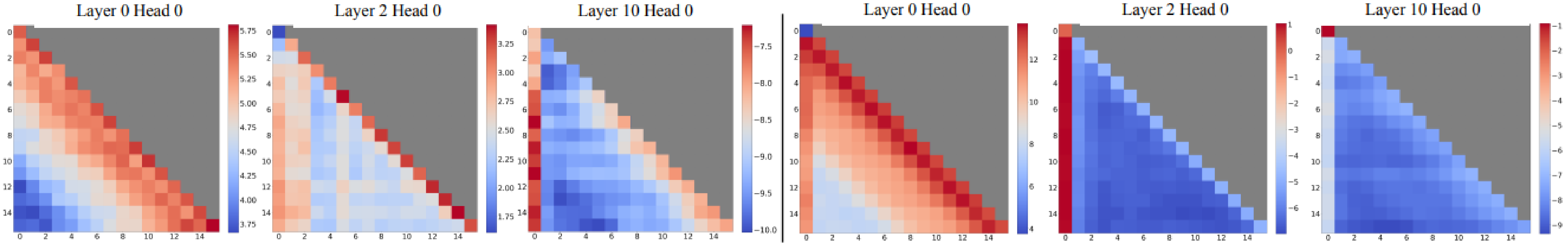
- Experiments showed most models require first four tokens to recover normal text quality (perplexity)
- Cached KV values use positions values 1 to L, regardless of if current tokens are well beyond the Lth token
- Using RoPE or ALiBi, the base token embeddings can be cached, then the positional information applied on top of the base embeddings
- Pre-training a model with a learned attention sink reduces requirement to a single attention sink



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

Pre-trained model attention patterns

- Attention on first token is more consistent in models pre-trained with an attention sink token



Pre-Trained without Sink Token

Pre-Trained with Sink Token

- These pre-trained models had similar performance on benchmarks when compared to models without attention sinks

Attention sinks conclusion

- Using attention sinks during decoding solves text generation quality after the context exceeds the max sequence length
  - This approach works with KV caching when using additive, RoPE, or ALiBi position embeddings
- There’s almost no downside, since sparing a few tokens is negligible
- Pre-training models to expect attention sinks reduces cost to just a single attention sink token
- Context length is still a problem for LLMs. This work adds to the growing work to understand transformers.

References

- H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models
  Zhenyu Zhang et al. (2023)
  https://arxiv.org/abs/2306.14048
- RoFormer: Enhanced Transformer with Rotary Position Embedding
  Jianlin Su et al. (2021)
  https://arxiv.org/abs/2104.09864
- Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation
  Ofir Press et al. (2021)
  https://arxiv.org/abs/2108.12409