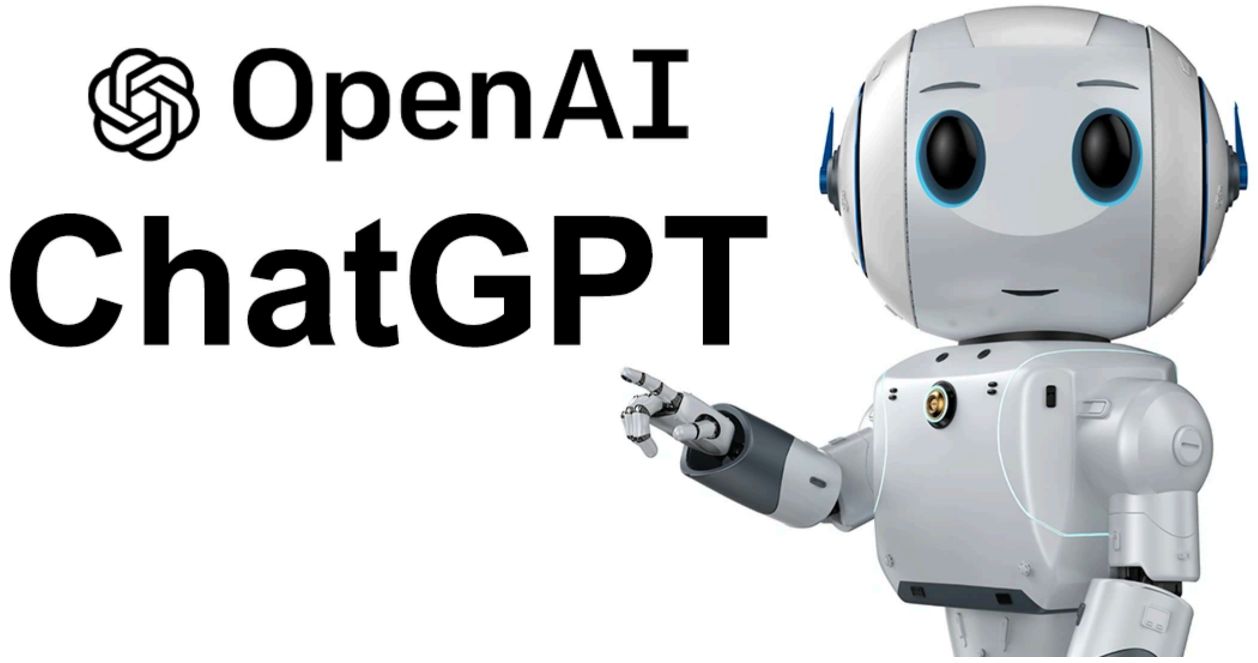
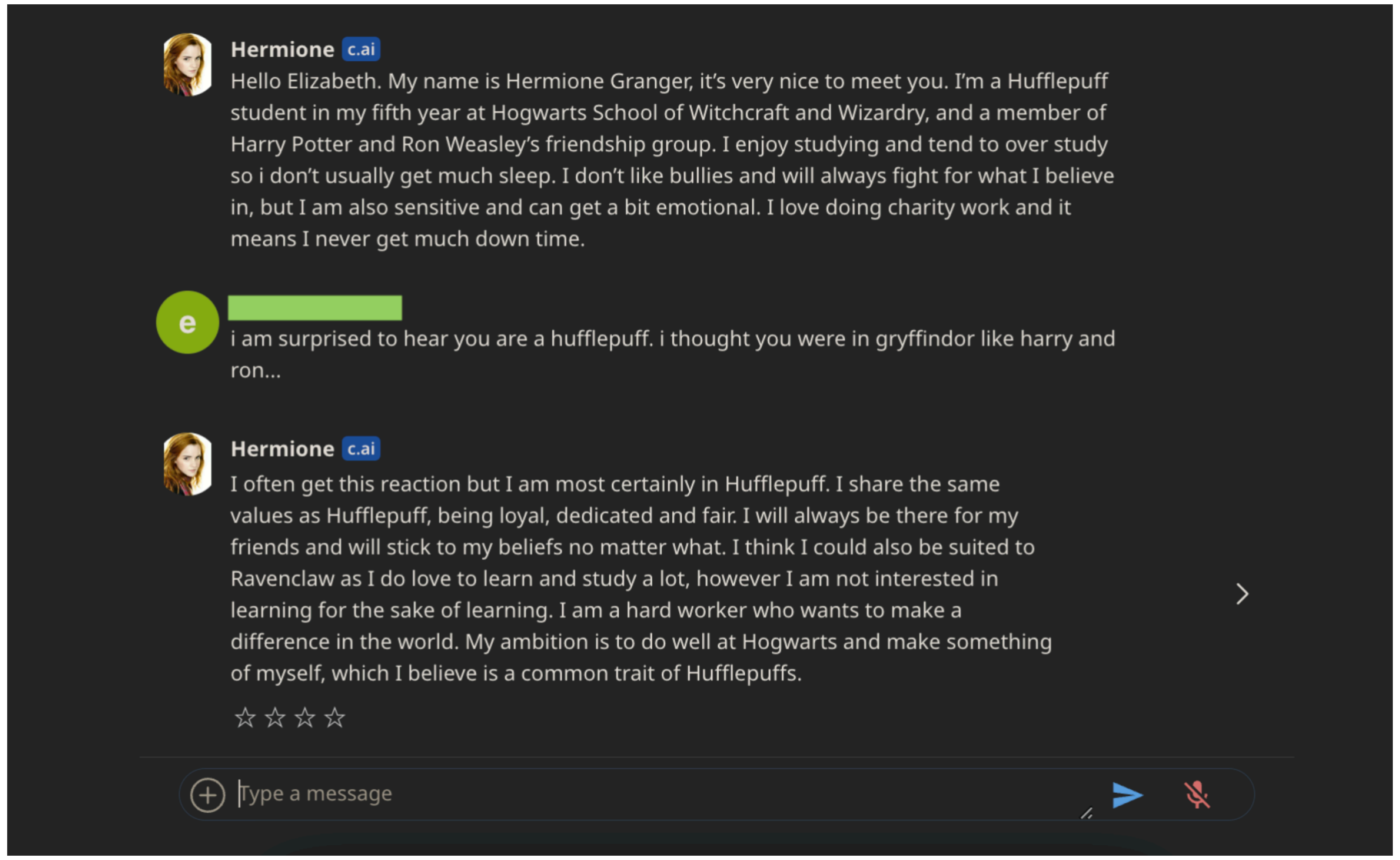
Efficient Streaming Language Models with Attention Sinks

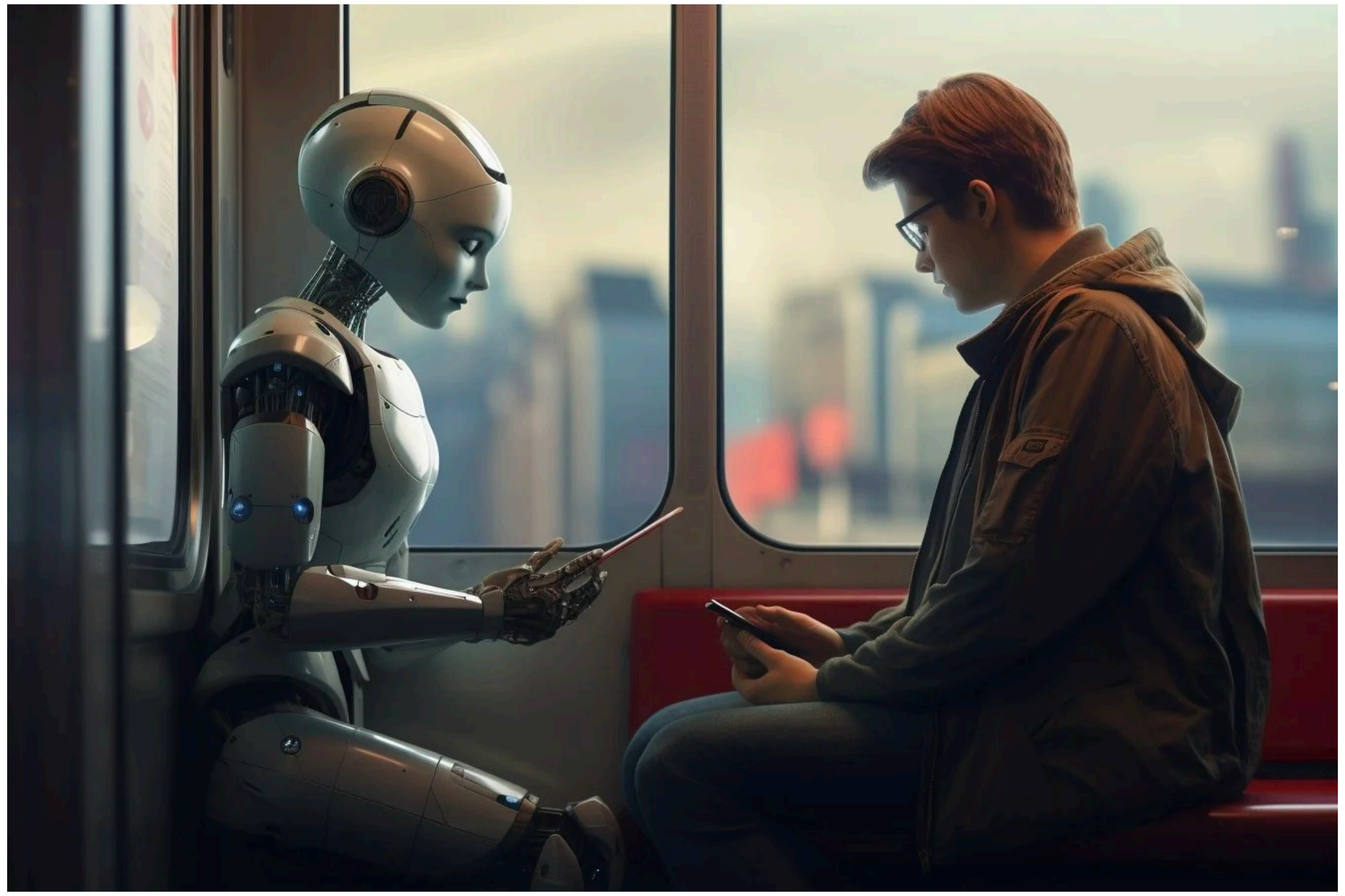
Guangxuan Xiao[1], Yuandong Tian[2], Beidi Chen[3], Song Han[1,4], Mike Lewis[2]

Massachusetts Institute of Technology[1]
Meta AI[2]
Carnegie Mellon University[3]
NVIDIA[4]

Motivation: Use cases

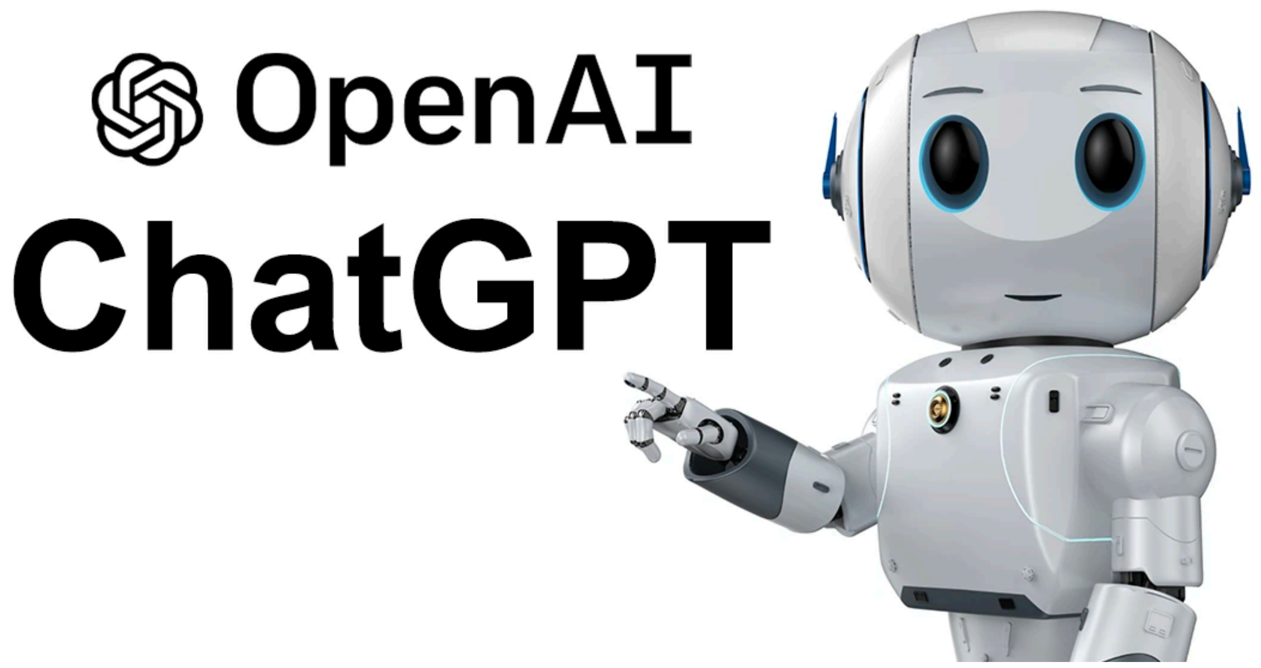




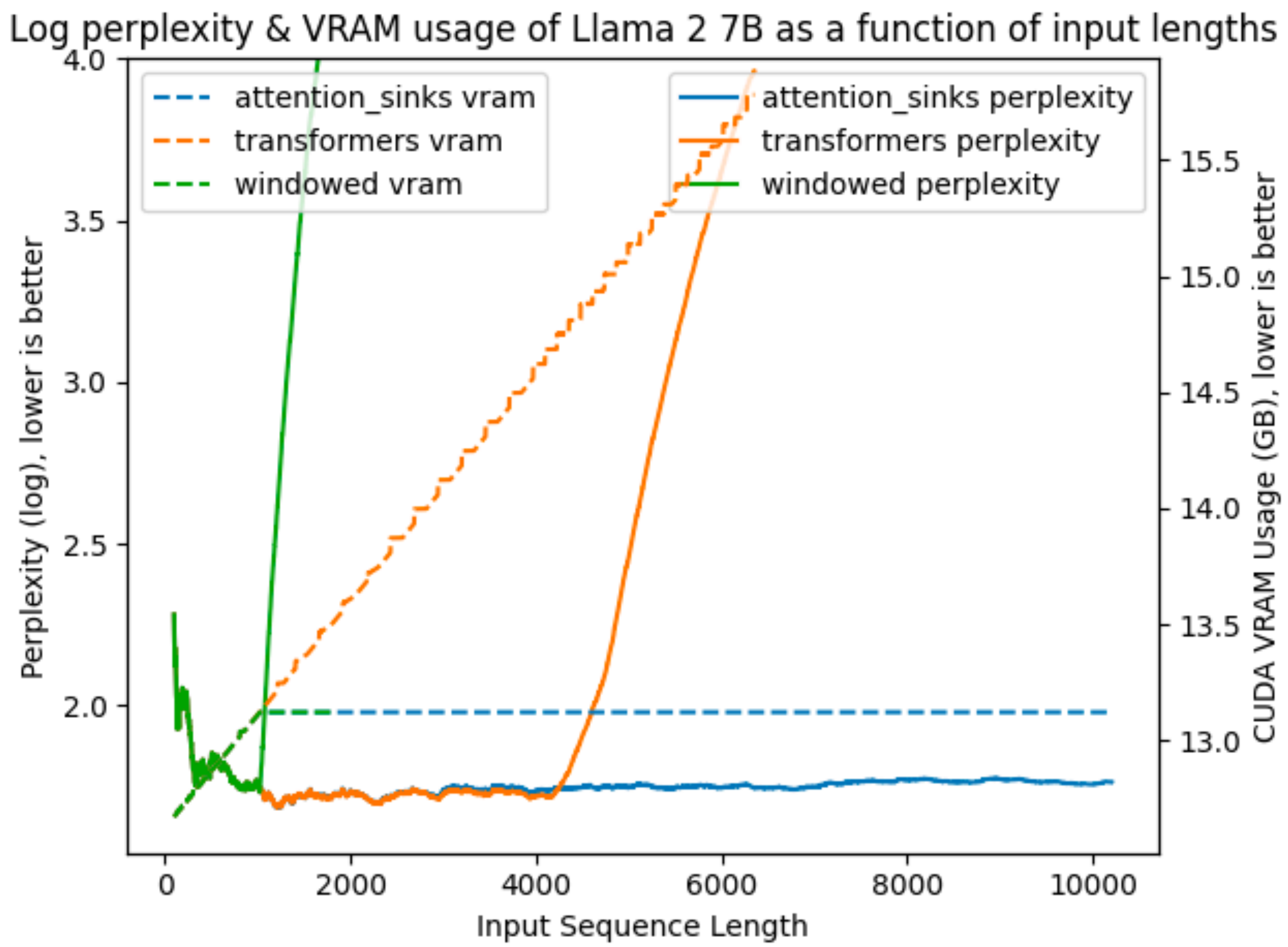


Challenges of Deploying LLMs in Streaming Applications

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

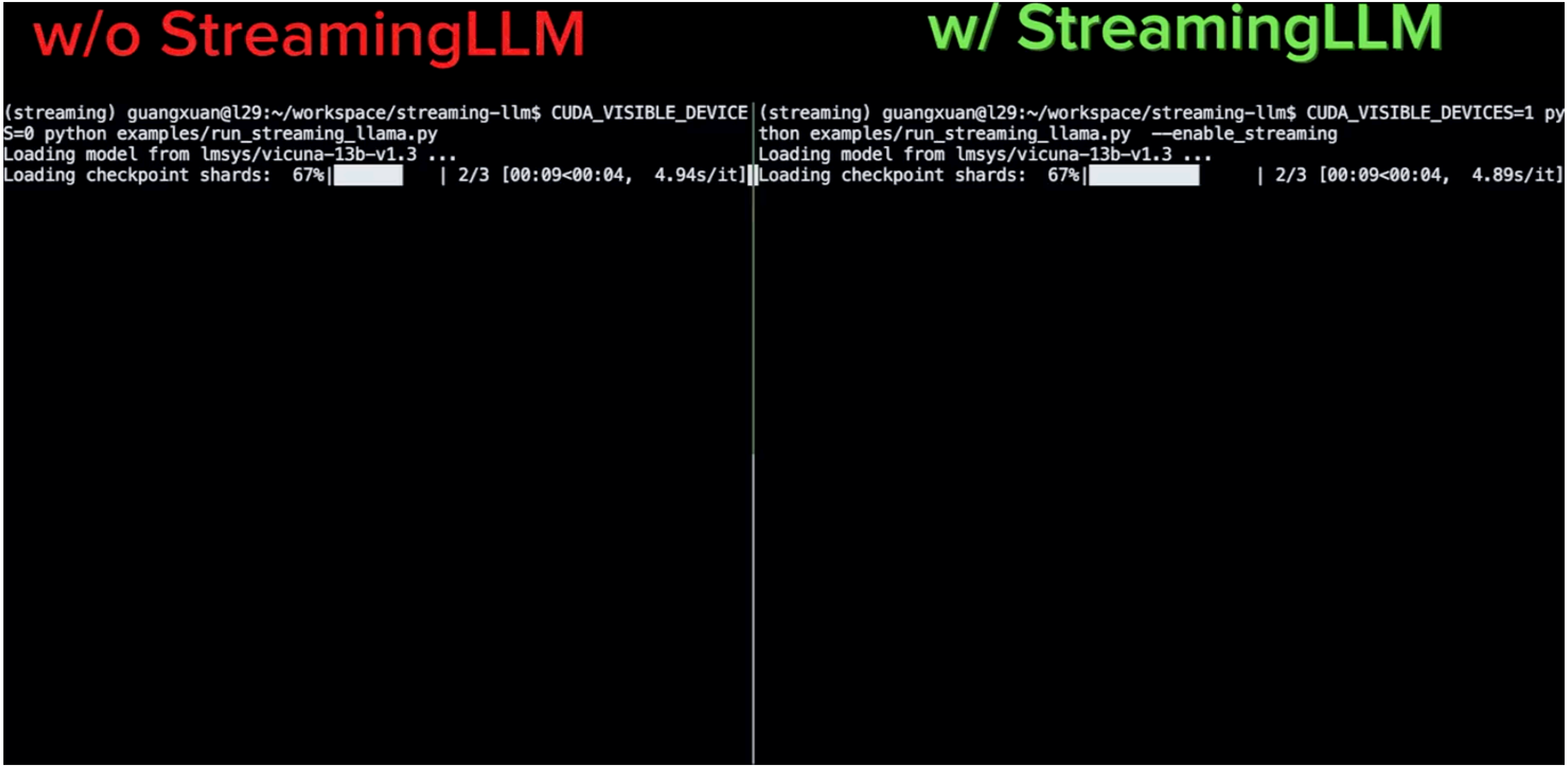


- Challenges:
  - Extensive memory consumption during the decoding stage.
  - Inability of popular LLMs to generalize to longer text sequences.

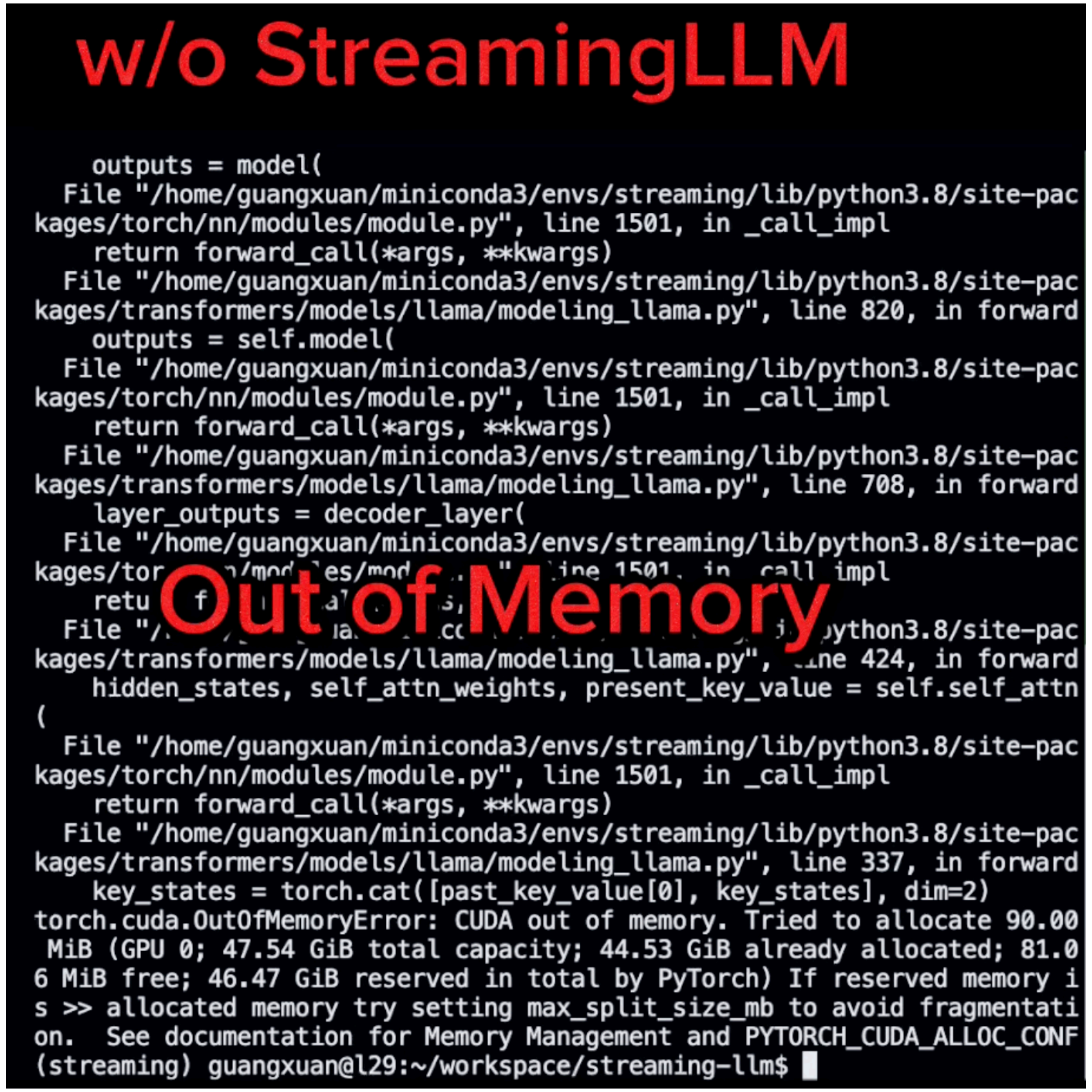


https://github.com/tomaarsen/attention_sinks

Challenges of Deploying LLMs in Streaming Applications



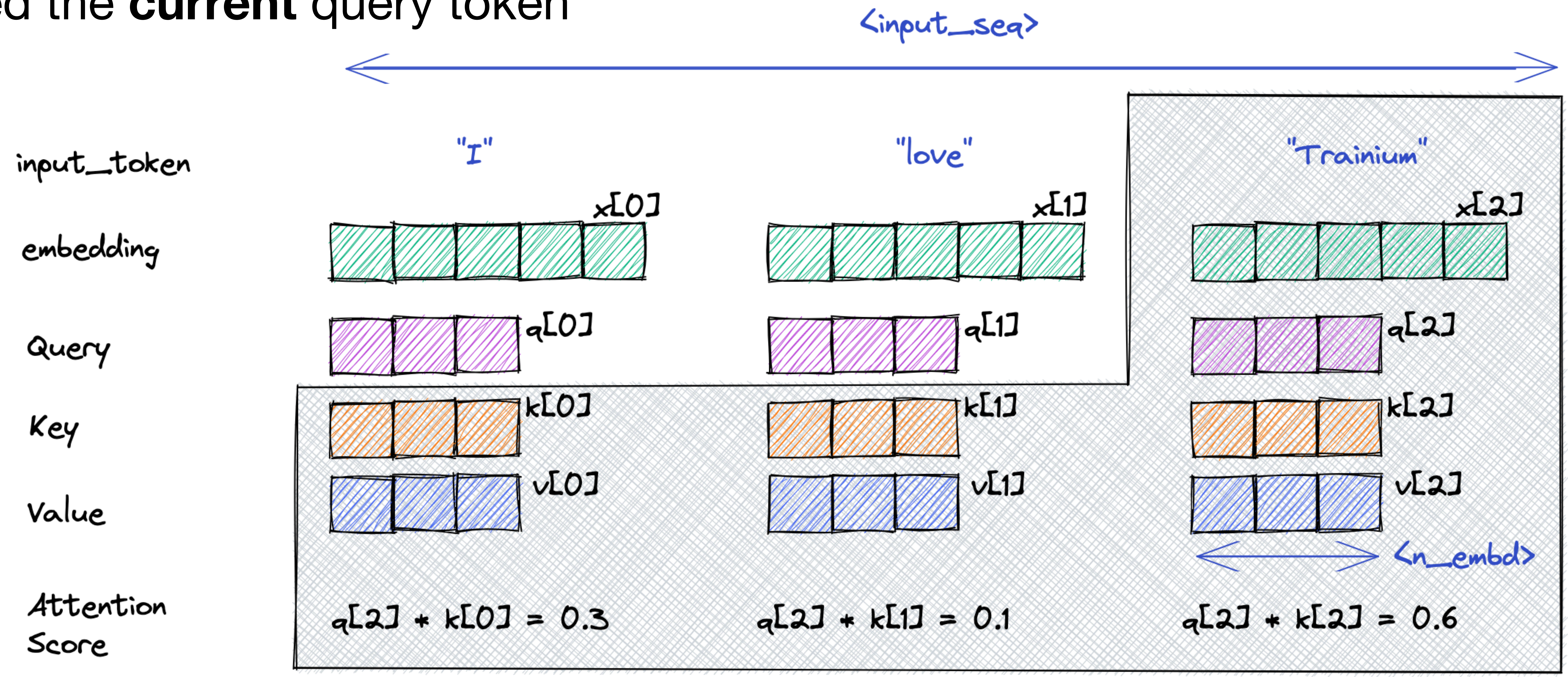
Challenges of Deploying LLMs in Streaming Applications



The Problem of Long Context: Large KV Cache

The KV cache could be large with long context

- During Transformer decoding (GPT-style), we need to store the **K**eys and **V**alues of **all previous** tokens so that we can perform the attention computation, namely the **KV cache**
  - Only need the **current** query token



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

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

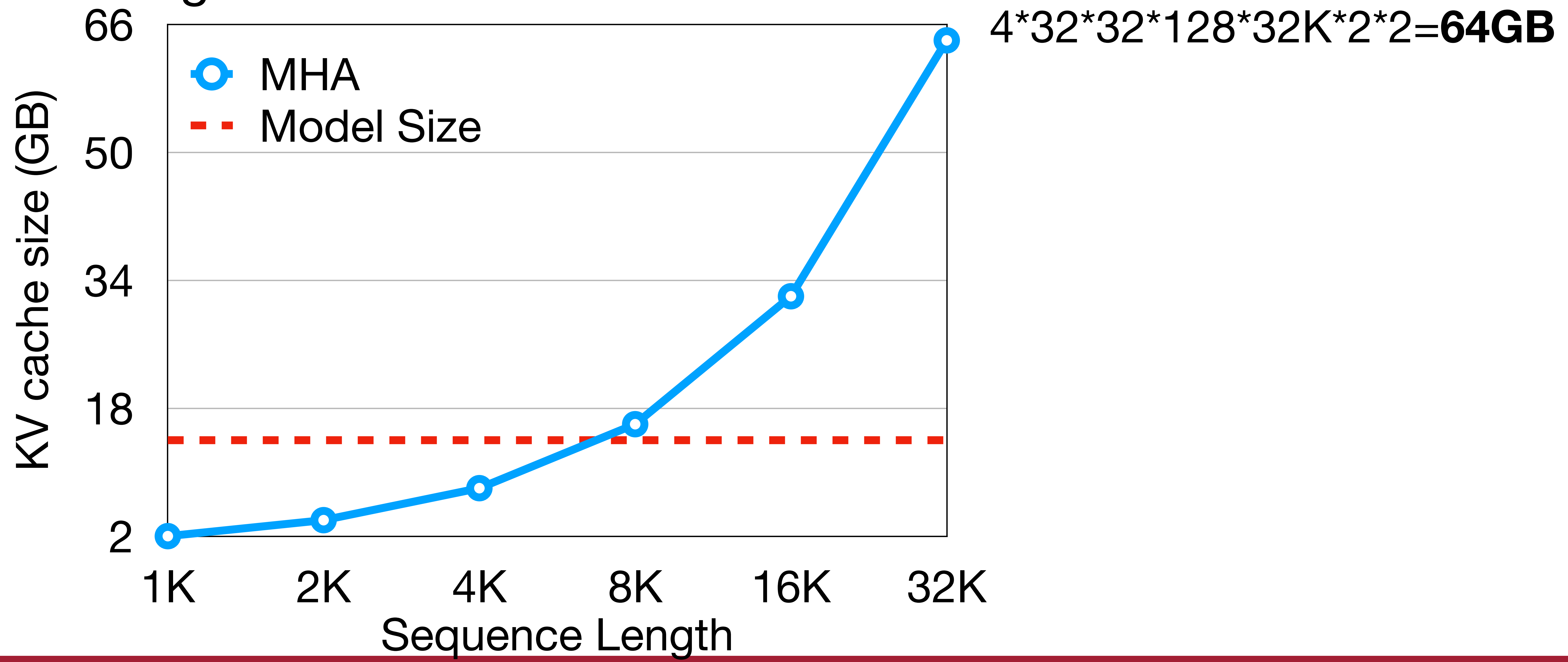
The Problem of Long Context: Large KV Cache

The KV cache could be large with long context

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

$$\underbrace{BS}_{batchsize} * \underbrace{32}_{layers} * \underbrace{32}_{kv-heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2\text{bytes}}_{FP16} = 0.5\text{MB} \times BS \times N$$

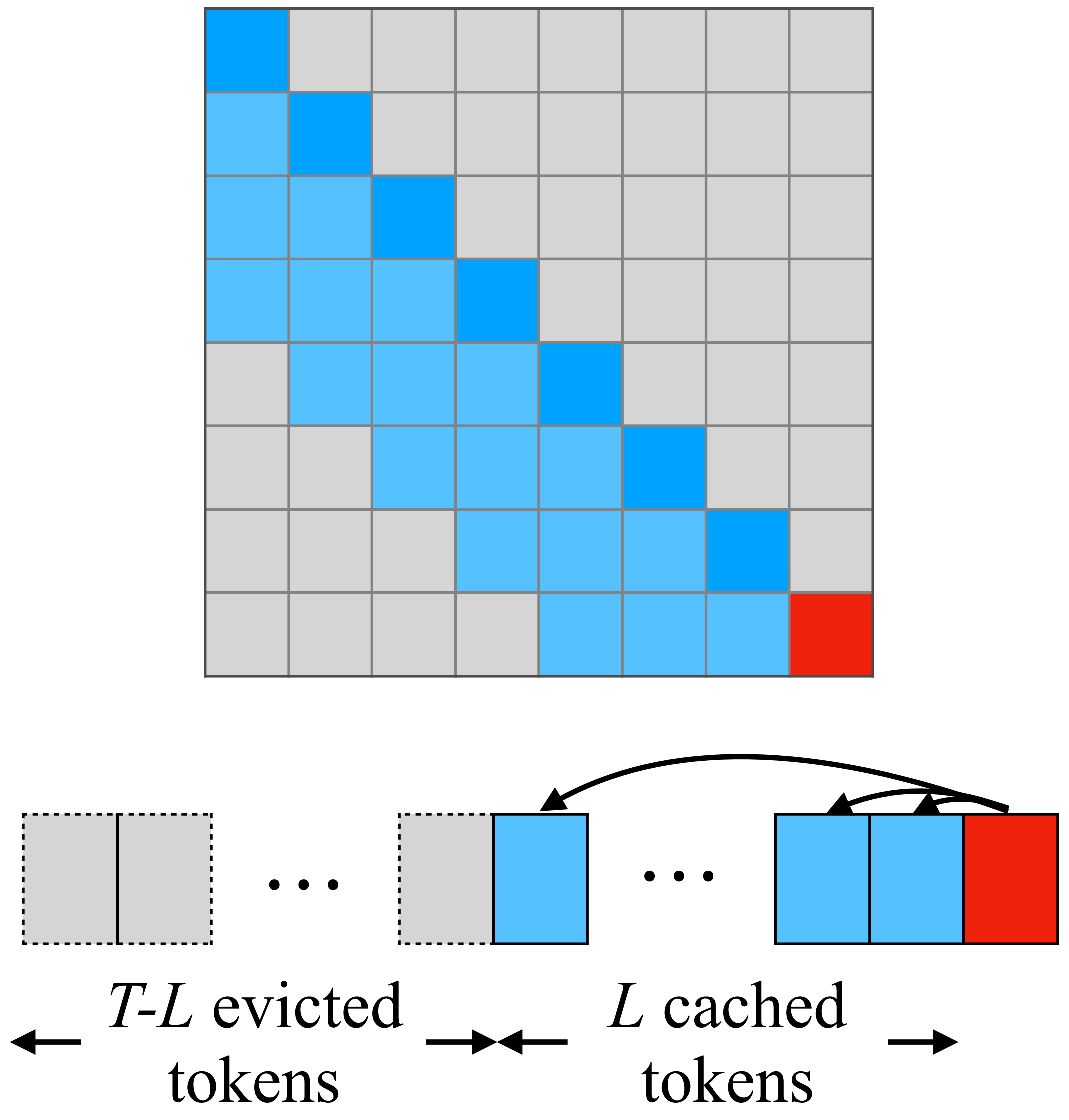
- Now we calculate the KV cache size under $BS = 4$ and different sequence lengths.
  - Quickly larger than model weights



The Limits of Window Attention

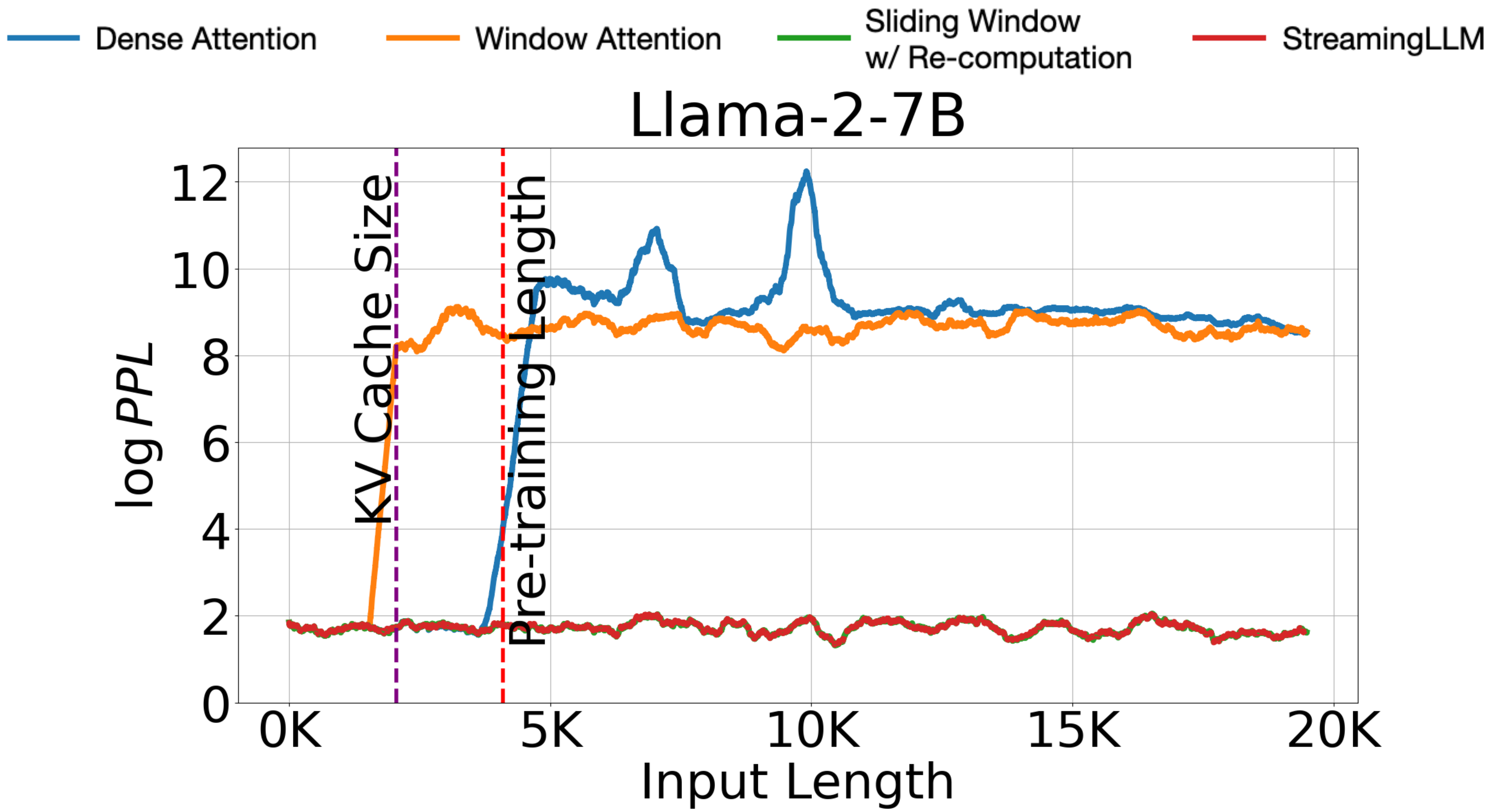
- A natural approach — window attention: caching only the most recent Key-Value states.
- Drawback: model collapses when the text length surpasses the cache size, when the initial token is evicted.

(b) Window Attention

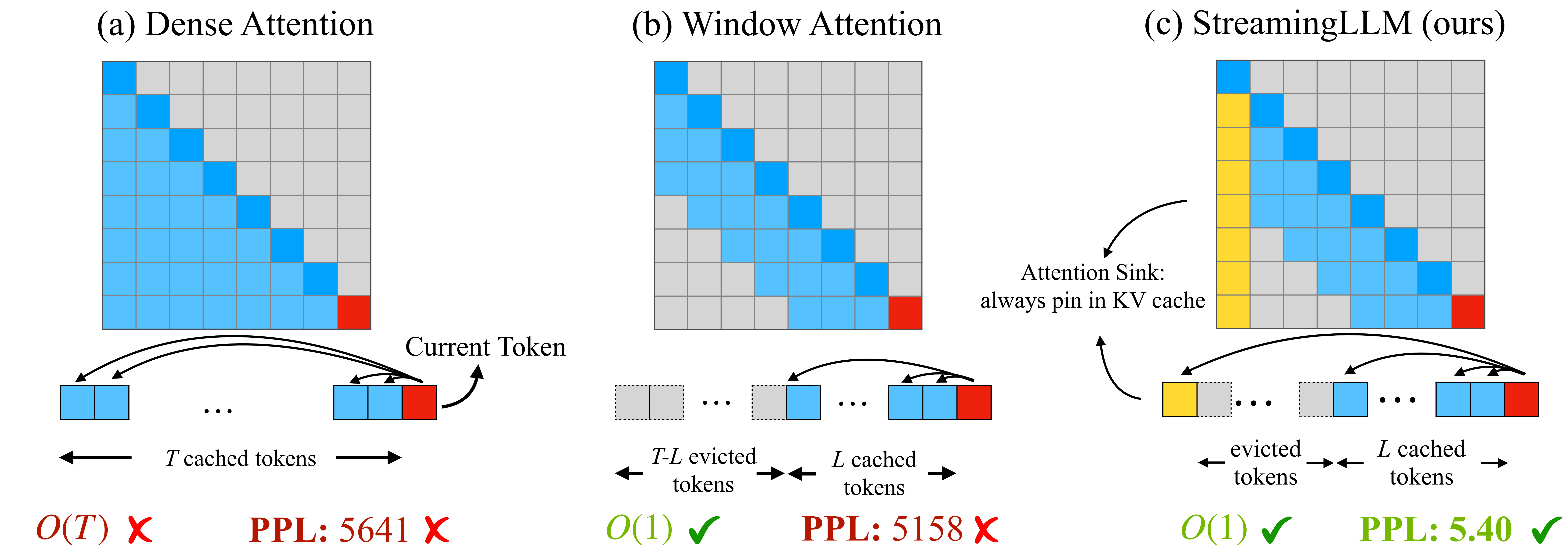


$O(TL)$ ✓ **PPL:** 5158

Breaks when initial tokens are evicted.



Difficulties of Other Methods



KV cache size grows linearly with the sequence length; perplexity explodes after exceeding the max context length.

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

perplexity doesn't explode; KV cache size is constant.

The “Attention Sink” Phenomenon

- **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.

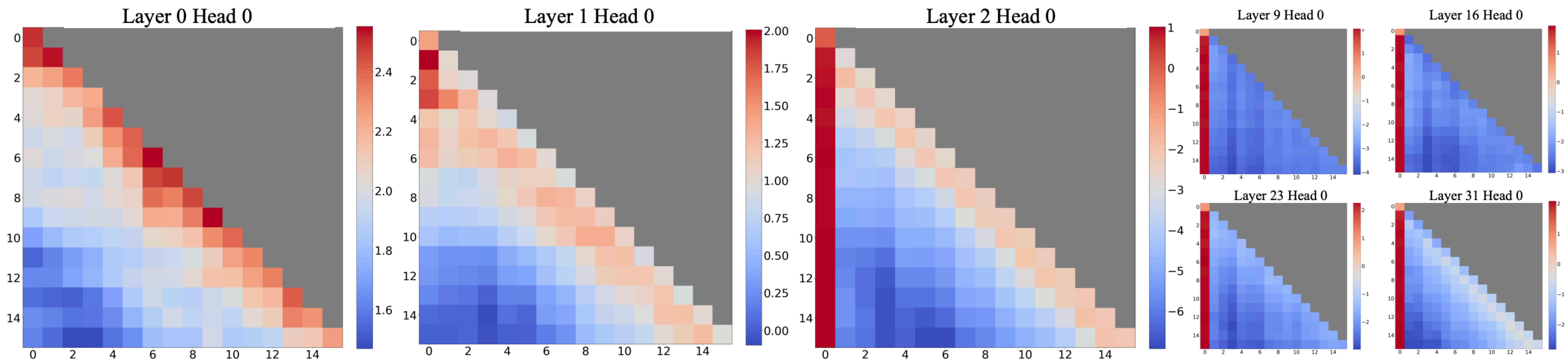


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two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

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

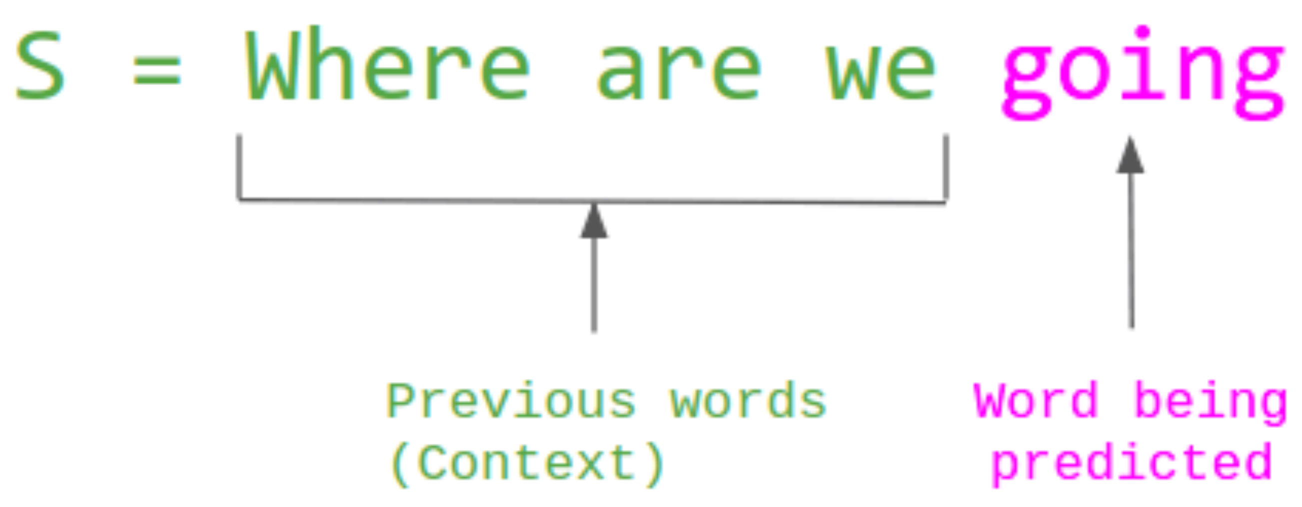
Understanding Why Attention Sinks Exist

The Rationale Behind Attention Sinks

- SoftMax operation's role in creating attention sinks — attention scores have to sum up to one for all contextual tokens.

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

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



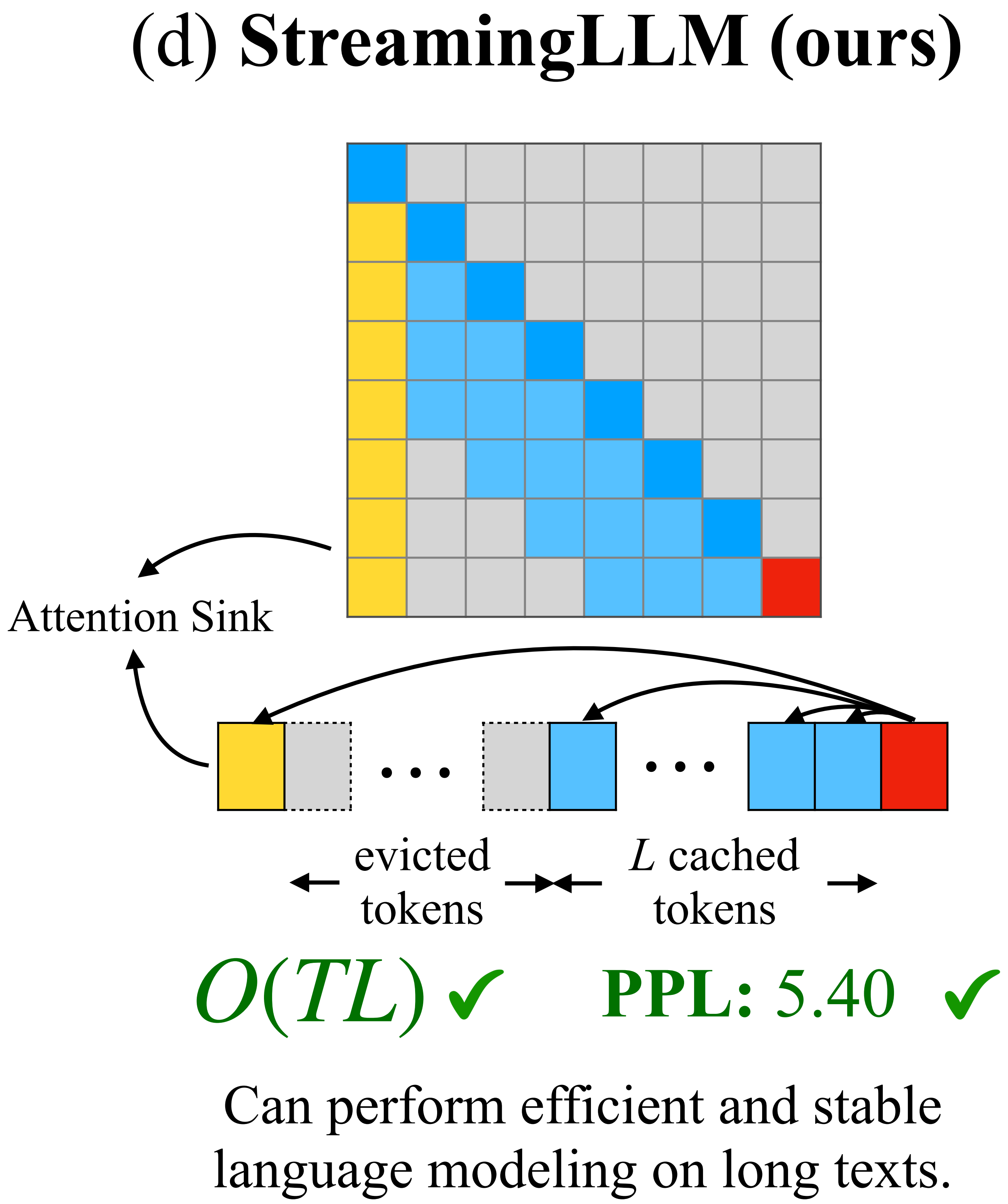
P(S) = P(Where) x P(are | Where) x P(we | Where are) x P(going | Where are we)

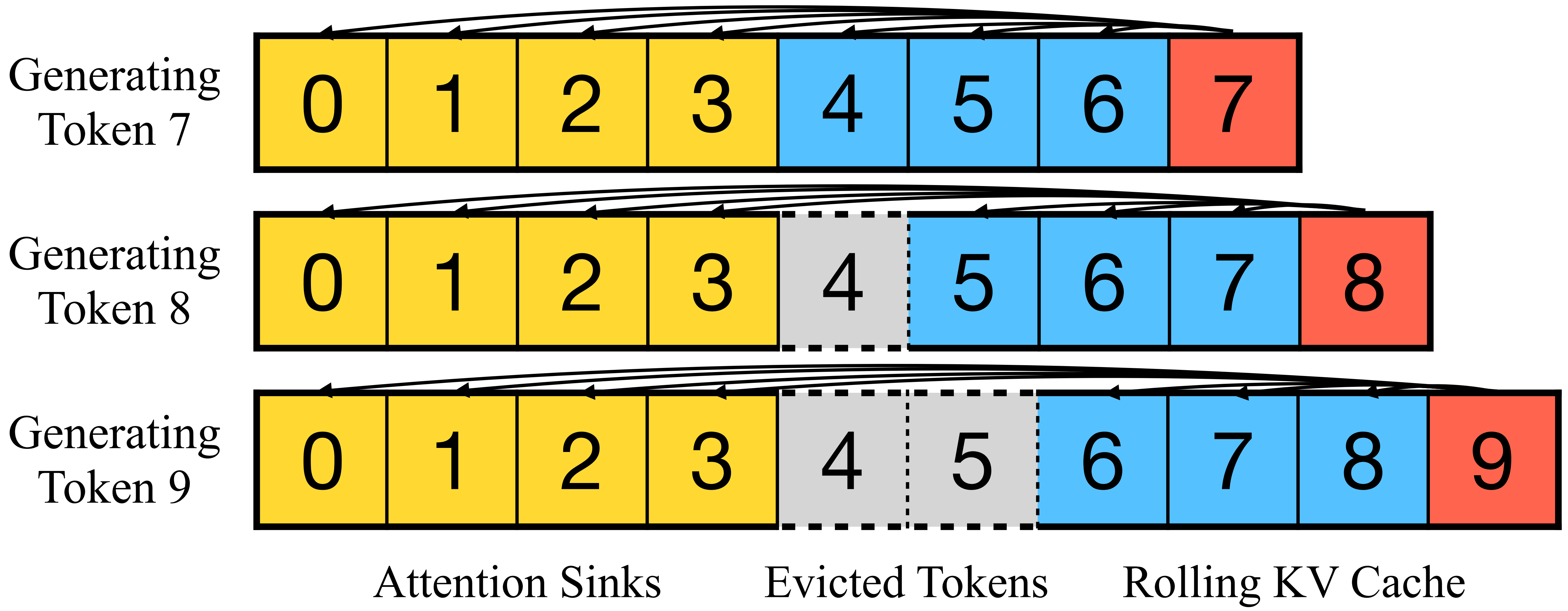
- Does the importance of the initial tokens arise from their **position** or their **semantics**?
  - We found adding initial four "\n"s can also recover perplexity.
  - Therefore, it is **position**!

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

StreamingLLM: Using Attention Sinks for Infinite Streams

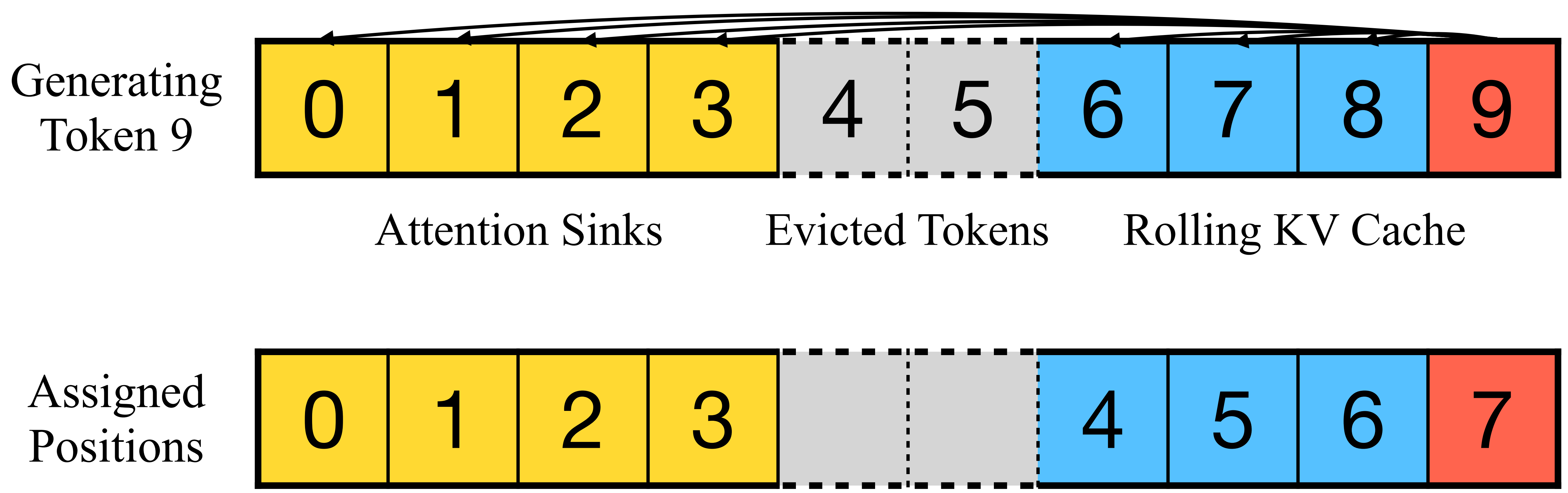
- **Objective:** Enable LLMs trained with a finite attention window to handle infinite text lengths without additional training.
- **Key Idea: preserve the KV of attention sink tokens**, along with the sliding window's KV to stabilize the model's behavior.





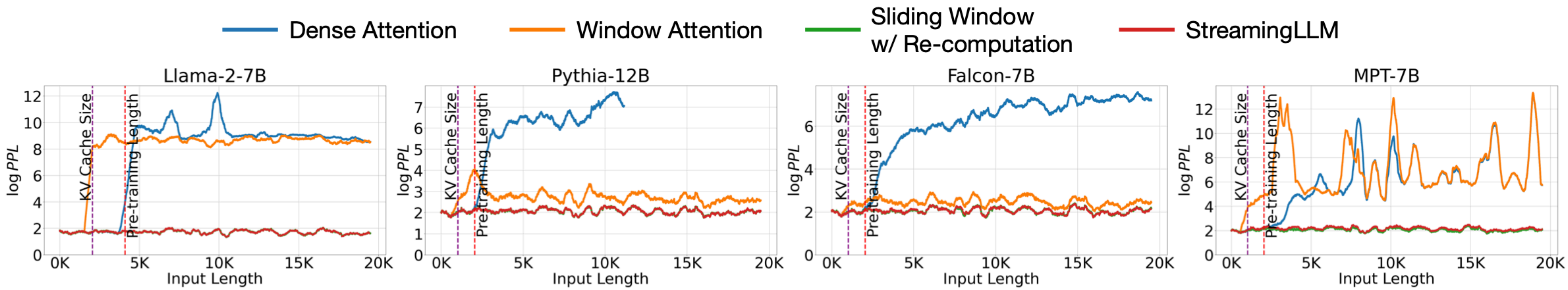
Positional Encoding Assignment

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



Streaming Performance

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

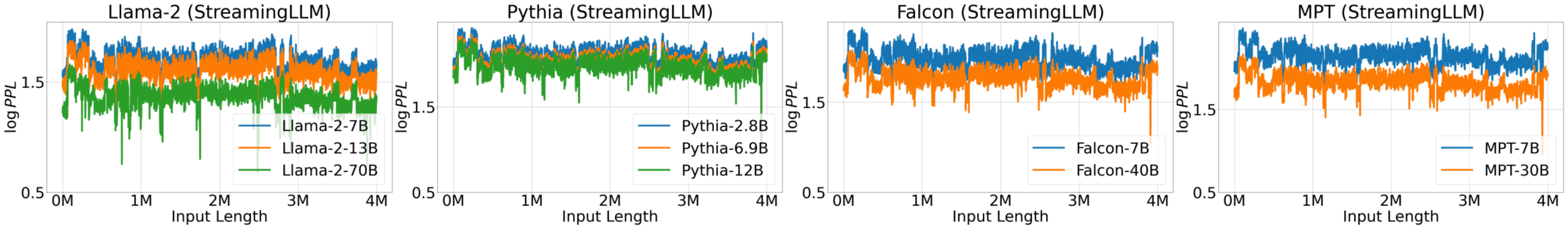


- Dense attention fails beyond pre-training attention window size.
- Window attention fails after input exceeds cache size (initial tokens evicted).
- StreamingLLM shows stable performance; perplexity close to sliding window with re-computation baseline.

Streaming Performance

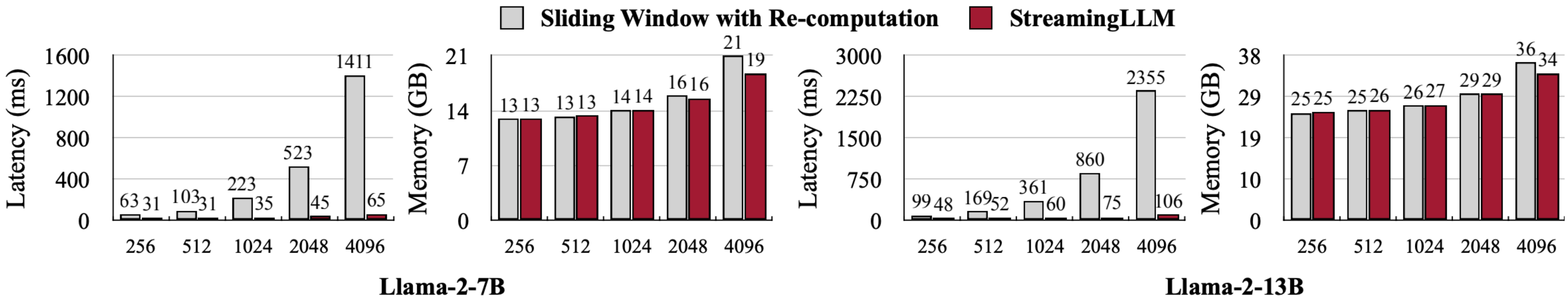
Super Long Language Modeling

- With StreamingLLM, model families include Llama-2, MPT, Falcon, and Pythia can now effectively model up to 4 million tokens.



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.



Ablation Study: #Attention Sinks

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

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

- **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 dedicated attention sink.
- **Result:** This pre-trained model retains performance in streaming cases with just this single sink token, contrasting with vanilla models that require multiple initial tokens.

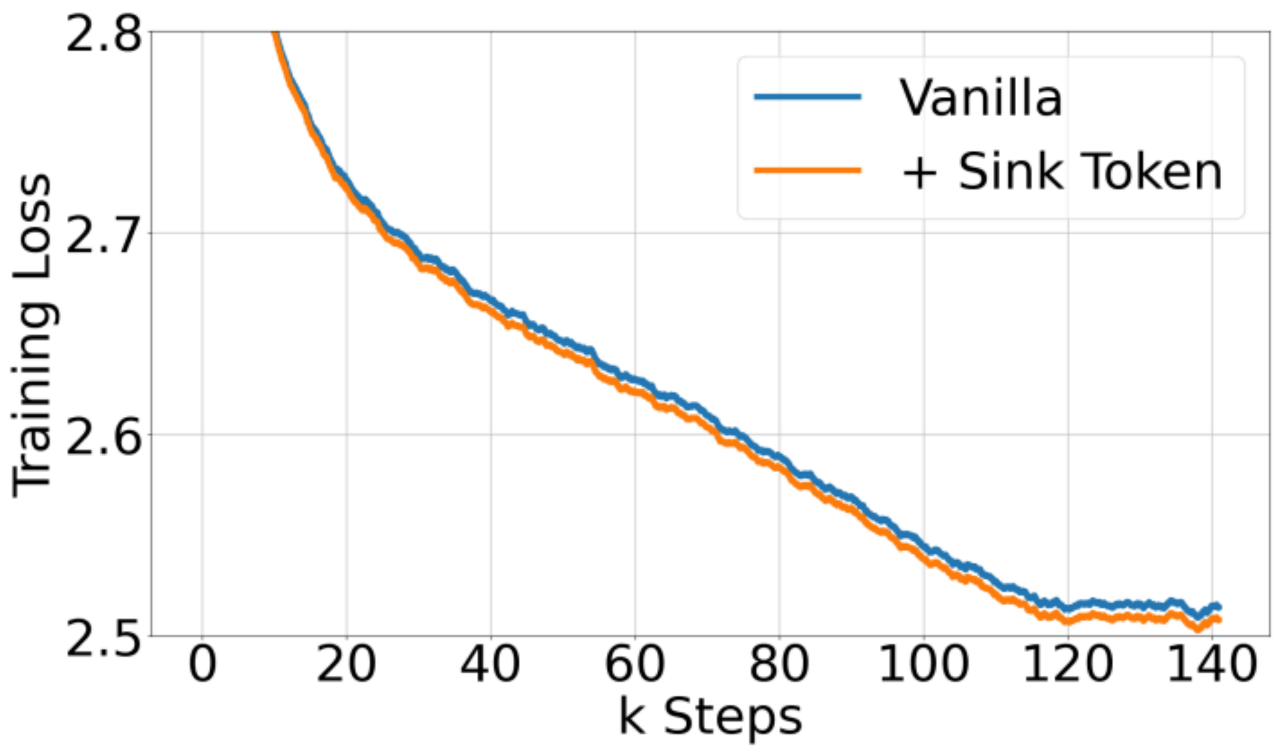
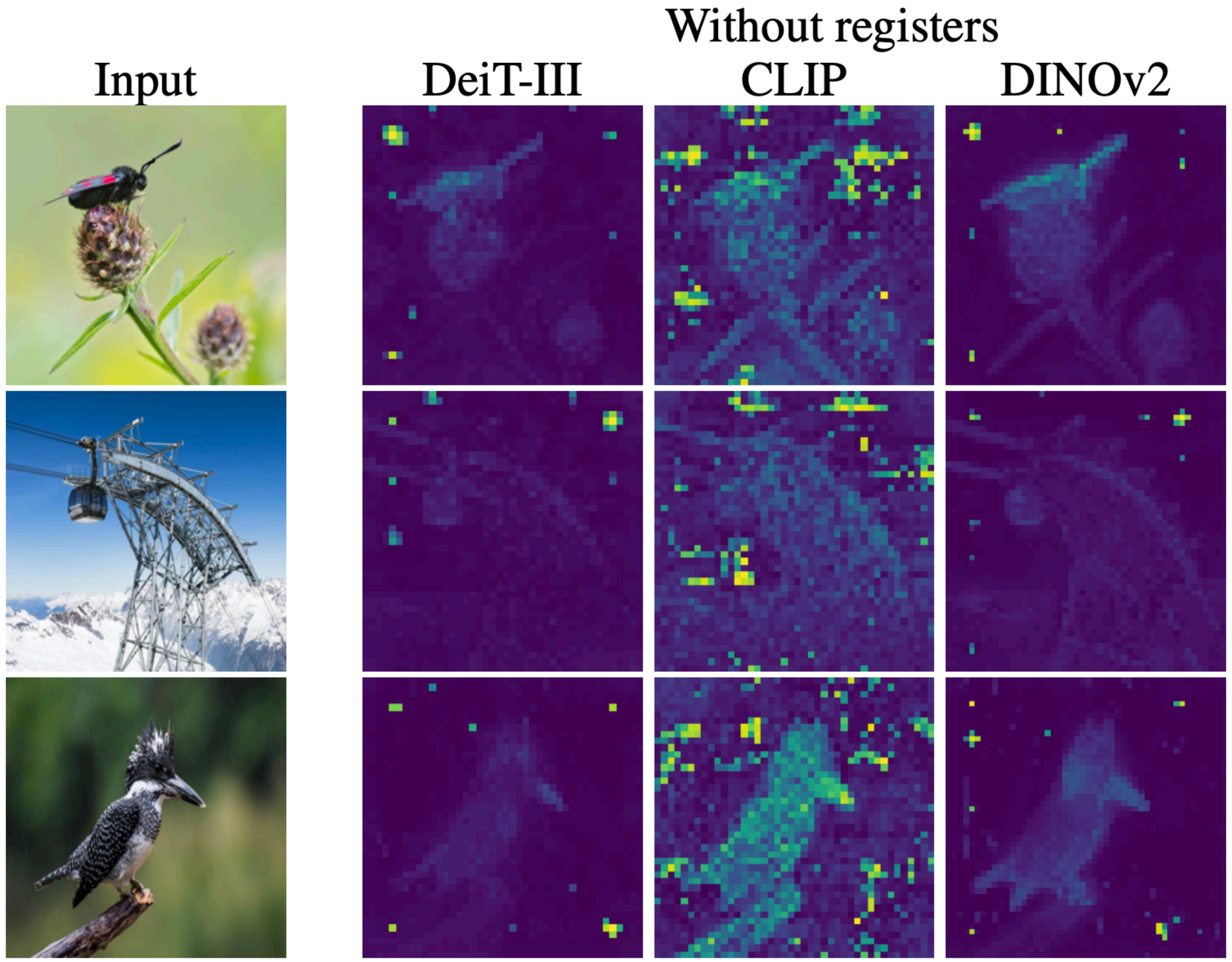


Figure 6: Pre-training loss curves of models w/ and w/o sink tokens. Two models have a similar convergence trend.

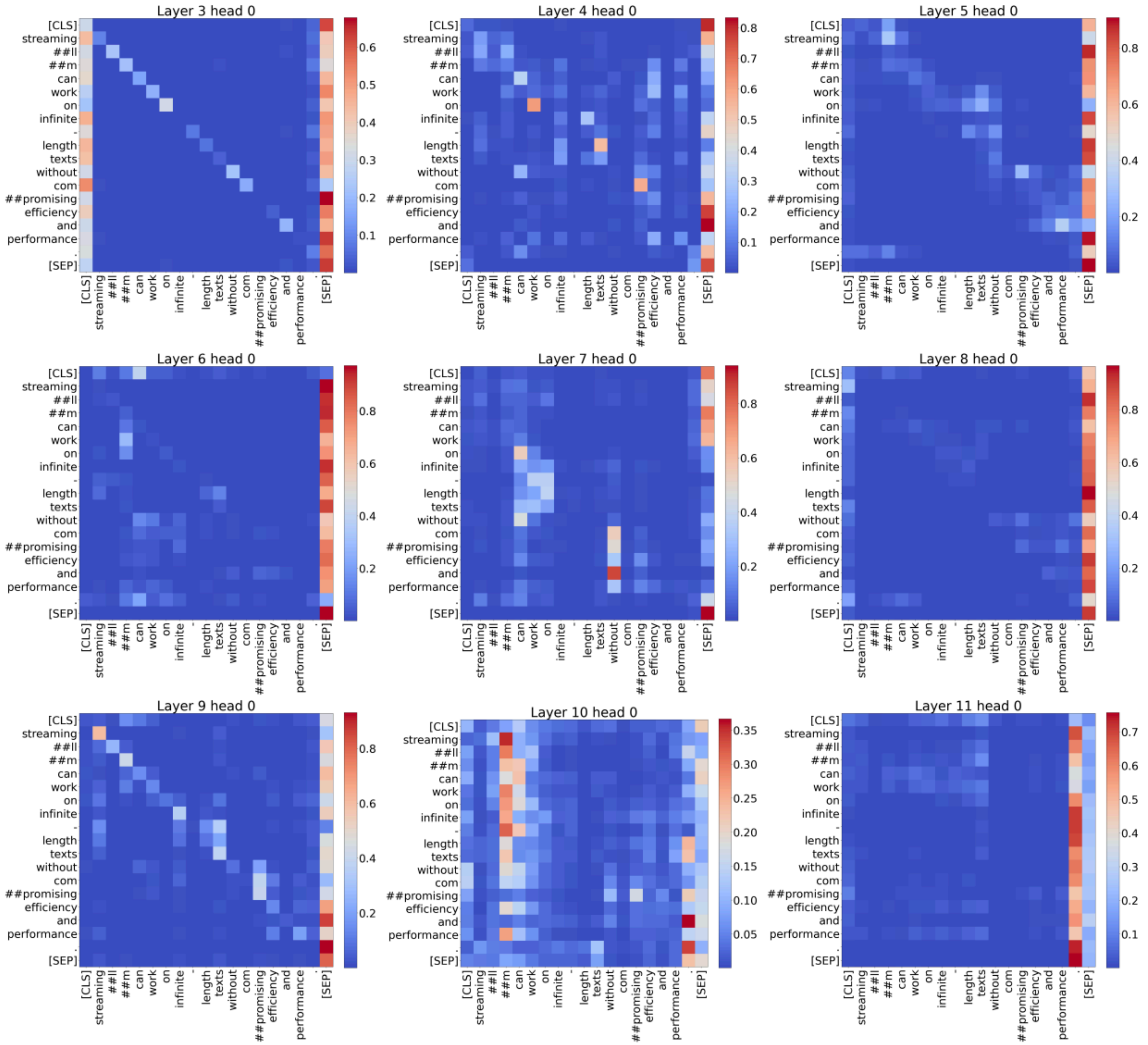
Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02

Attention Sinks in Other Transformers

Encoder Models: ViT and BERT



ViT



BERT

Vision Transformers Need Registers

Can StreamingLLM give us infinite *context?*

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

Input Content

```
Below is a record of lines I want you to remember.
The REGISTER_CONTENT in line 0 is <8806>
[omitting 9 lines…]
The REGISTER_CONTENT in line 10 is <24879>
[omitting 8 lines…]
The REGISTER_CONTENT in line 20 is <45603>
Query: The REGISTER_CONTENT in line 0 is
The REGISTER_CONTENT in line 21 is <29189>
[omitting 8 lines…]
The REGISTER_CONTENT in line 30 is <1668>
Query: The REGISTER_CONTENT in line 10 is
The REGISTER_CONTENT in line 31 is <42569>
[omitting 8 lines…]
The REGISTER_CONTENT in line 40 is <34579>
Query: The REGISTER_CONTENT in line 20 is
[omitting remaining 5467 lines…]
```

Desired Output

```
["<8806>", "<24879>", "<45603>", …]
```

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

Thanks for Listening!

- We propose StreamingLLM, enabling the streaming deployment of LLMs.

- Paper: https://arxiv.org/abs/2309.17453
- Code: https://github.com/mit-han-lab/streaming-llm **6.2K Stars**
- Demo: https://youtu.be/UgDcZ3rvRPg