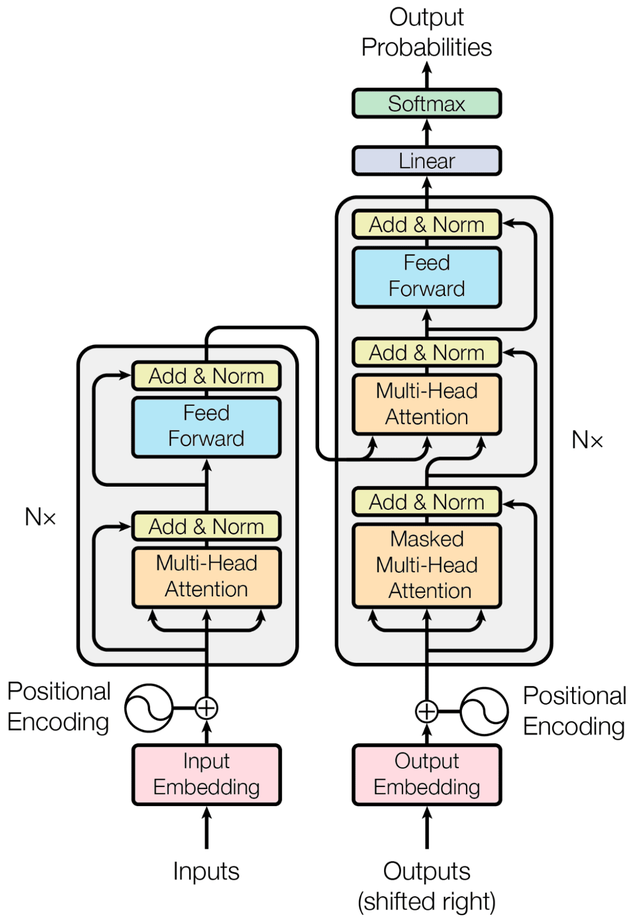
Efficient Large Language Models and Generative AI

Guangxuan Xiao

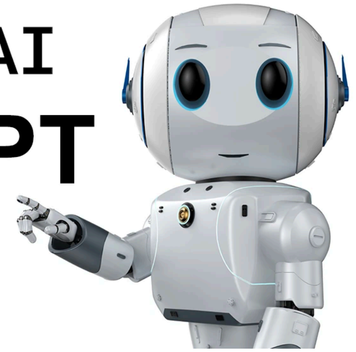
xgx@mit.edu
https://guangxuanx.com

Large Language Models (LLMs) are Powerful

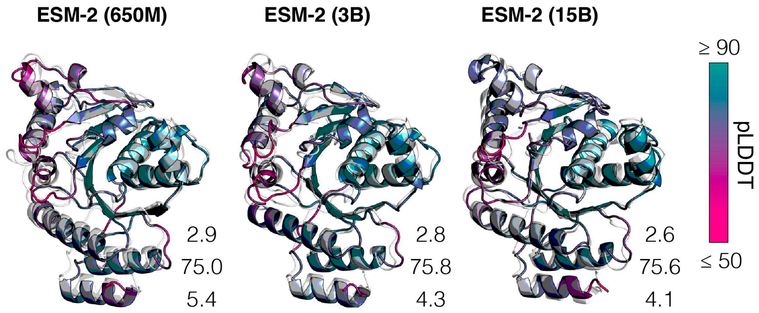


Transformer-based

OpenAI ChatGPT



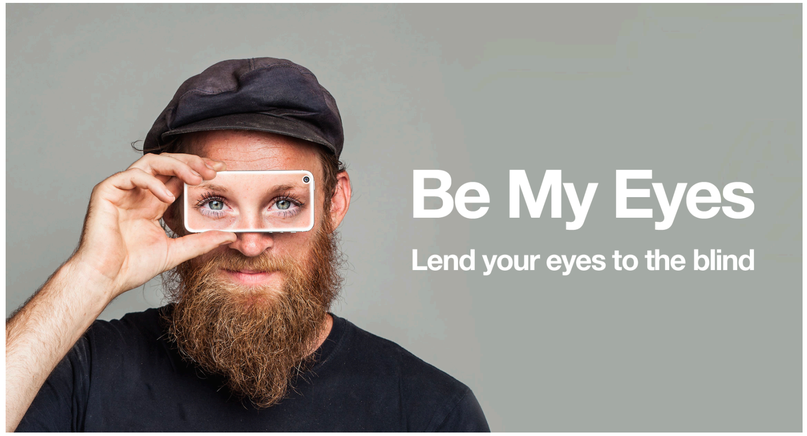
ChatBots



Scientific Discovery

GitHub Copilot

Software Development

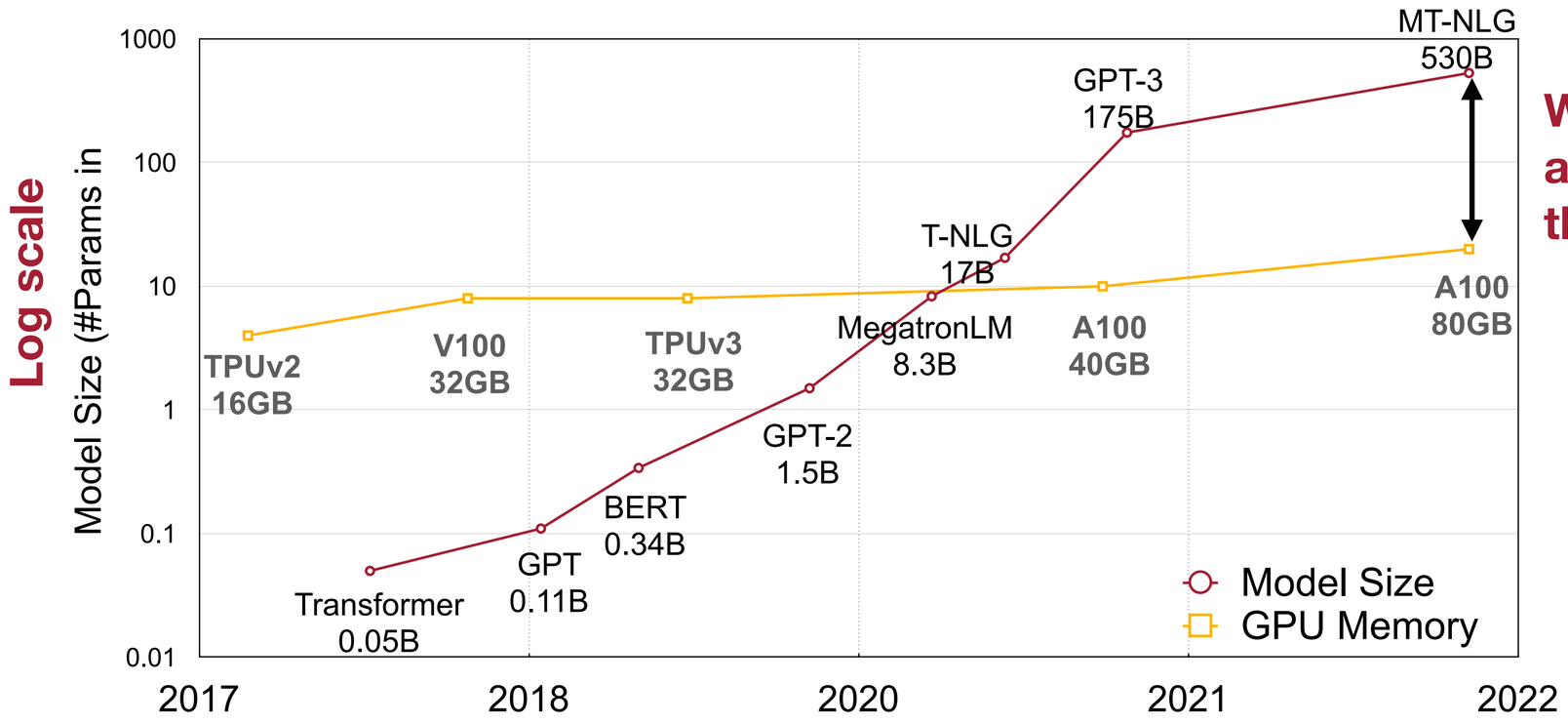


Disability Aid

LLM Models Outgrow Hardware Capacity

Reducing LLM serving cost is essential

- LLM sizes and computation are increasing exponentially.
- Domain-specific accelerator alone is not enough; we need efficient deployment algorithms to bridge the gap.



We need efficient algorithms to bridge the gap.

Content

Today I will introduce:

- **LLM Compression & Inference Speedup**
  - **SmoothQuant**: Accurate and Efficient Post-Training Quantization for Large Language Models (ICML 2023)

- **Efficient Application of LLMs for Extended Inputs**
  - Efficient Streaming Language Models with **Attention Sinks (StreamingLLM)** (ICLR 2024)

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models

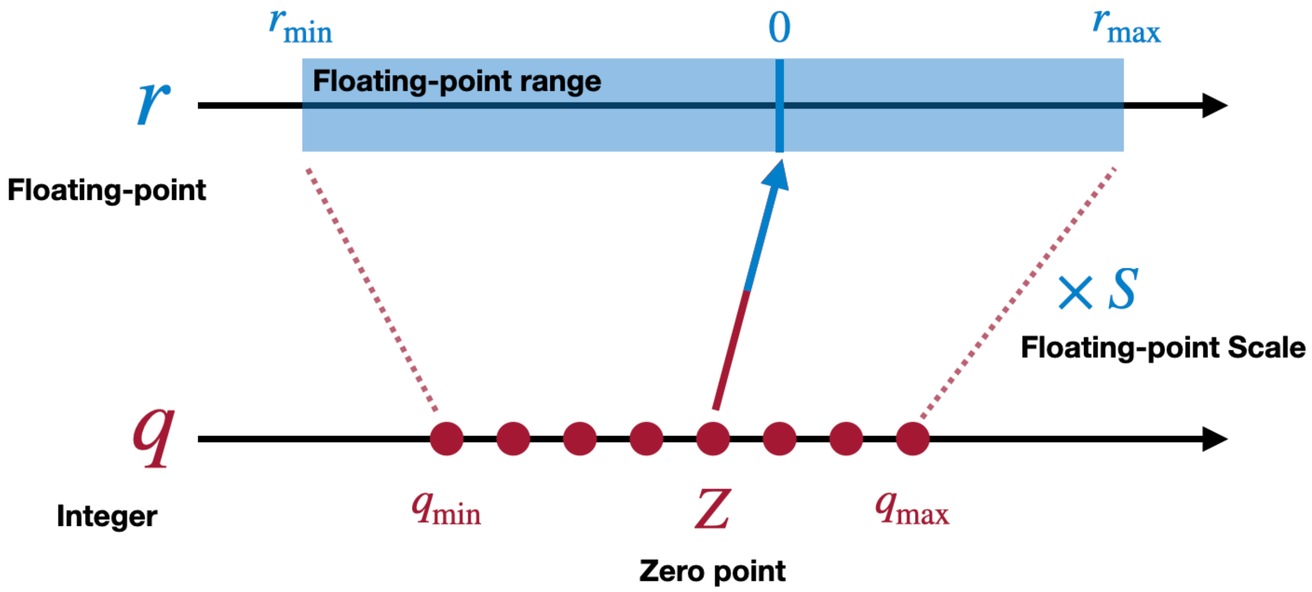
Guangxuan Xiao*[1], Ji Lin*[1], Mickael Seznec[2], Hao Wu[2], Julien Demouth[2], Song Han[1]

Massachusetts Institute of Technology[1]
NVIDIA[2]

Quantization Can Reduce Deployment Costs

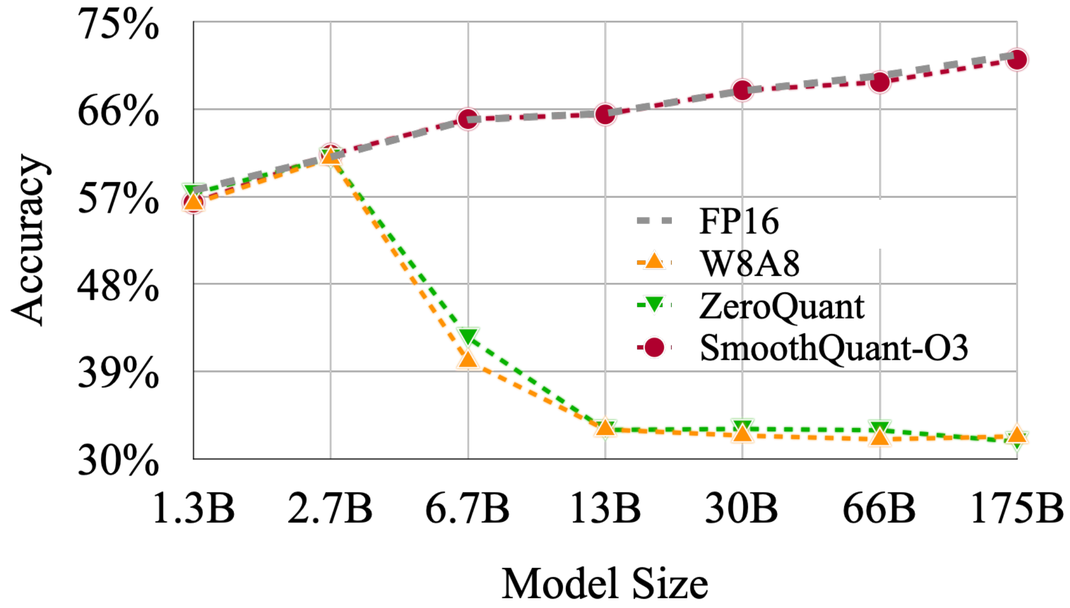
Quantization lowers the bit-width and improves efficiency

- Serving a 175B GPT-3 model at least requires:
  - FP16: 350GB memory ➡️ 5 x 80GB A100 GPUs
  - **INT8:** 175GB memory ➡️ **3 x 80GB A100 GPUs**



Naive Quantization Methods are Inaccurate

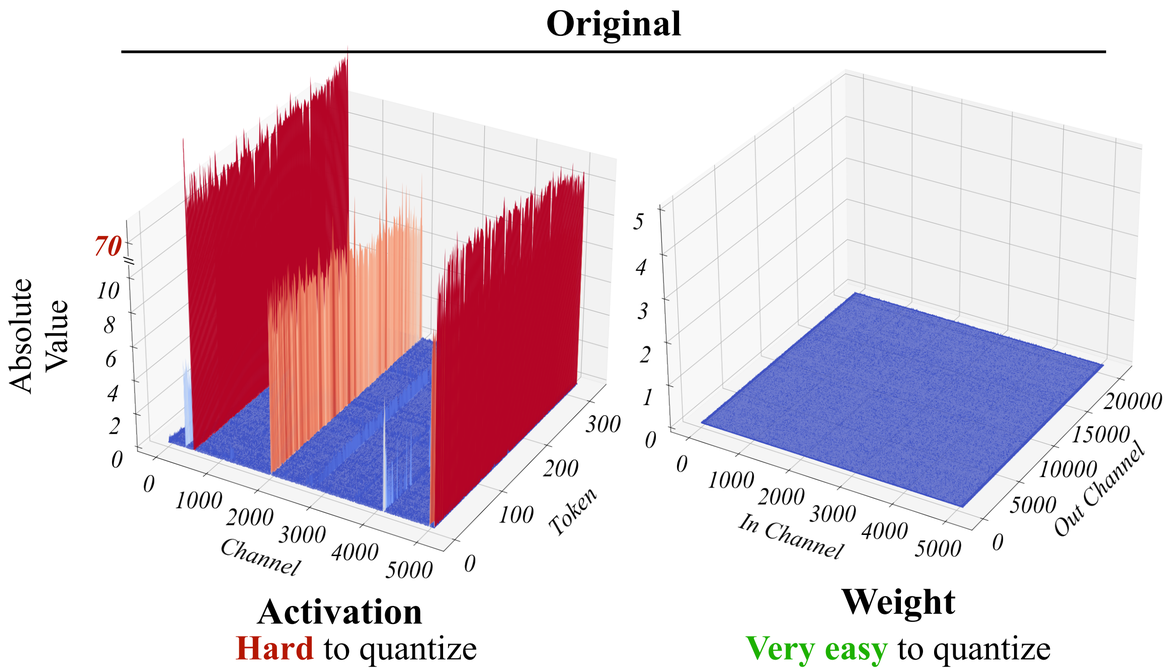
Activation outliers destroy quantized performance



- W8A8 quantization has been an industrial standard for CNNs, but not LLM. Why?
- Systematic outliers emerge in **activations** when we scale up LLMs beyond 6.7B. Traditional CNN quantization methods will destroy the accuracy.

Understanding the Quantization Difficulty of LLMs

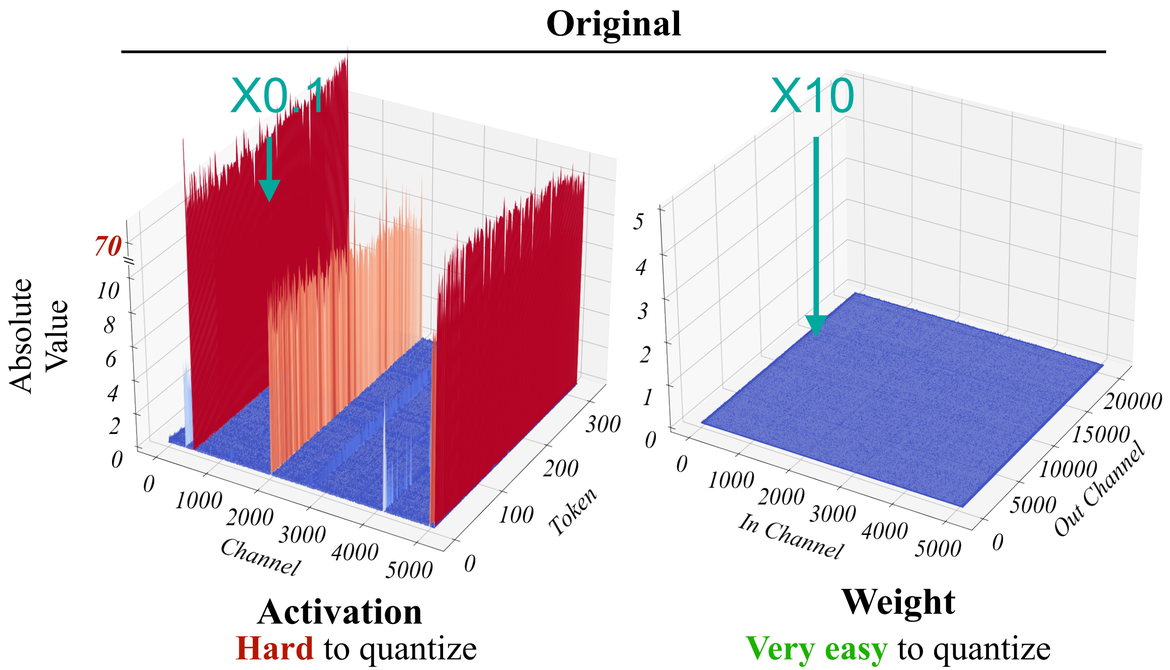
Smoothing activation to reduce quantization error



- Weights are easy to quantize, but activation is hard due to outliers

Understanding the Quantization Difficulty of LLMs

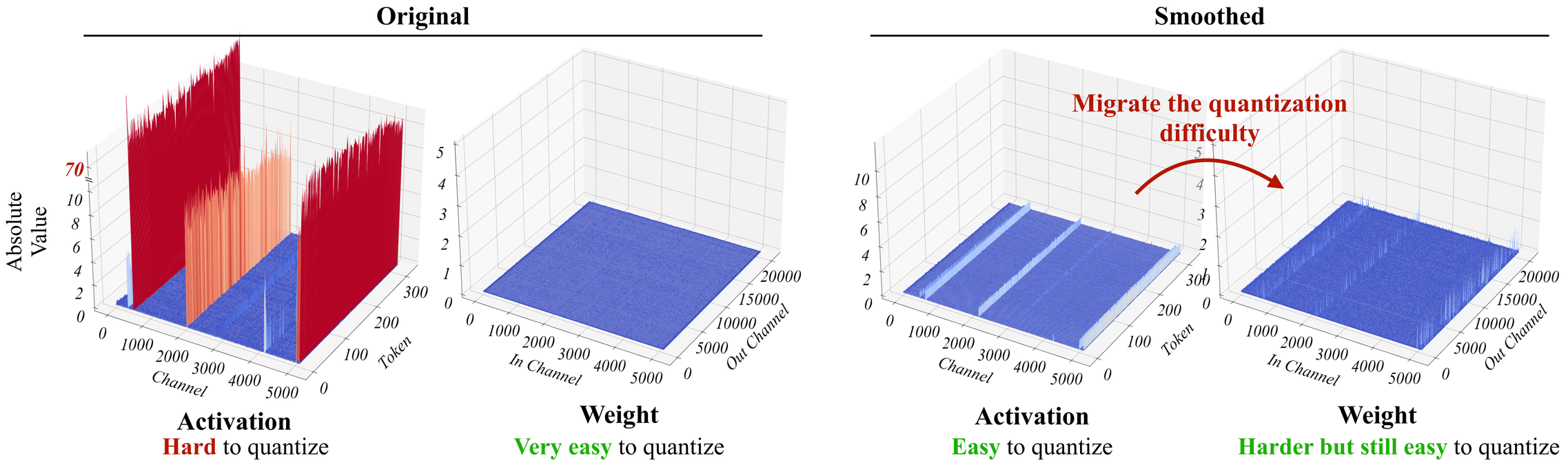
Smoothing activation to reduce quantization error



- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels

Understanding the Quantization Difficulty of LLMs

Smoothing activation to reduce quantization error

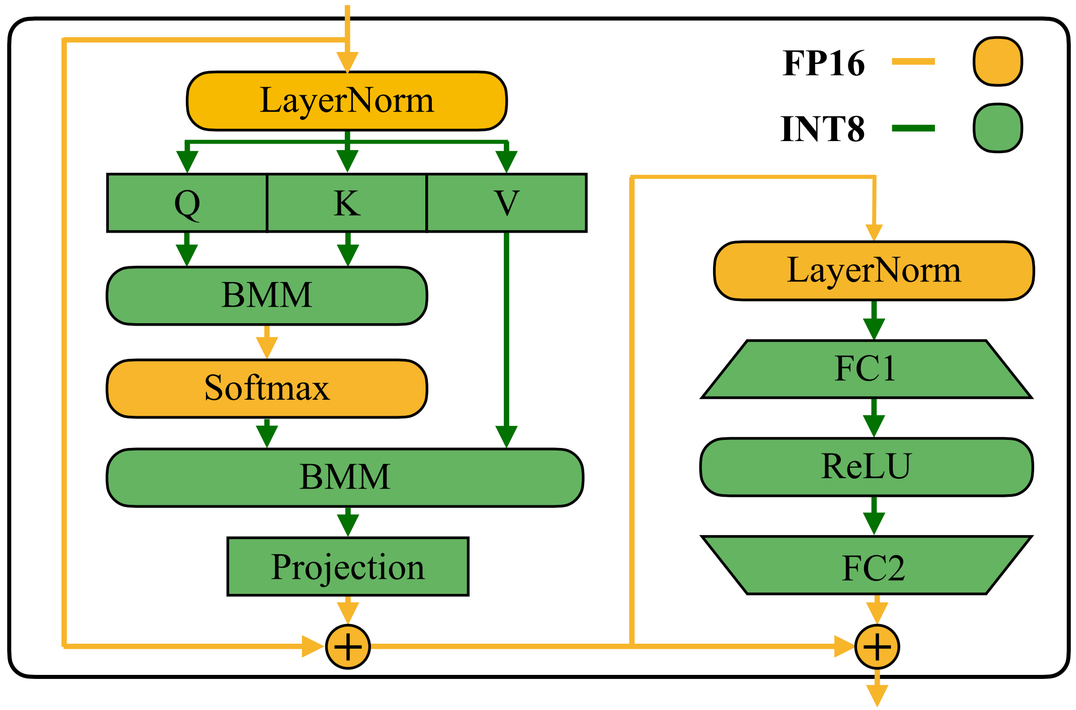


- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels
- Migrate the quantization difficulty from activation to weights, so both are easy to quantize

System Implementation

Efficient System Implementation

- We integrate SmoothQuant into FasterTransformer
- All compute-intensive operators (Linear, BMM) are quantized

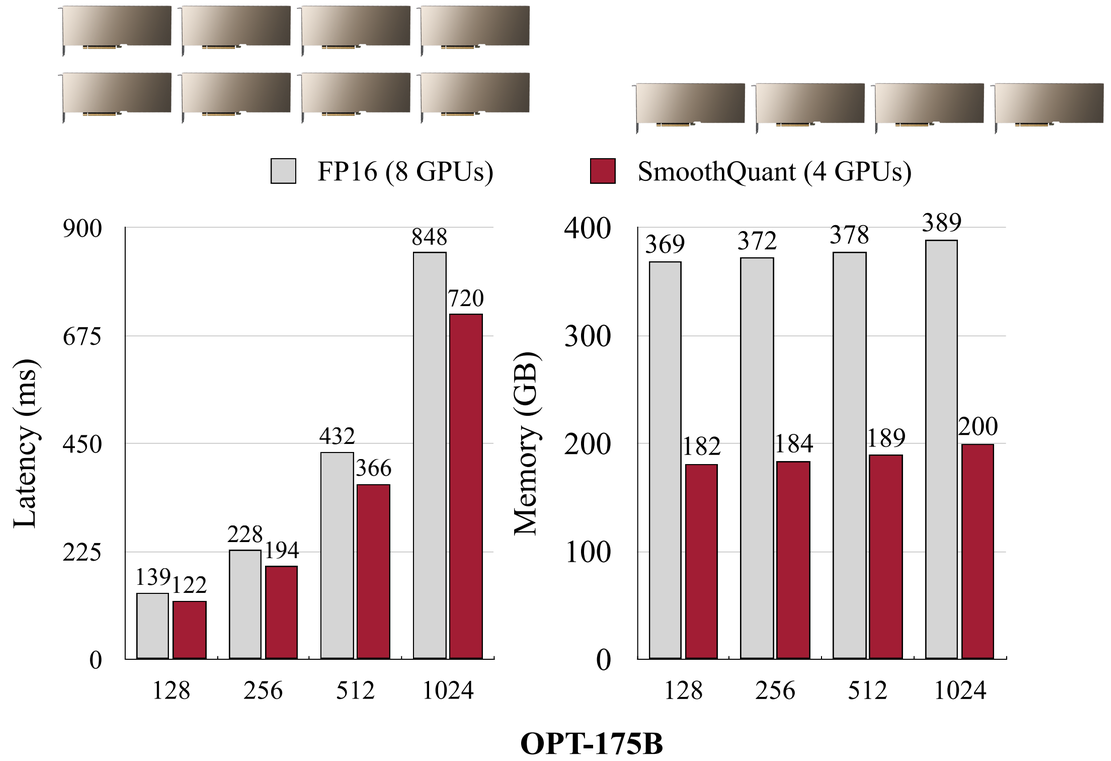


SmoothQuant is Accurate and Efficient

- SmoothQuant well maintains the accuracy without fine-tuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.

LAMBADA Accuracy

	OPT-175B	BLOOM-176B	GLM-130B
FP16	71.6%	68.2%	73.8%
SmoothQuant	71.2%	68.3%	73.7%



Industry & Community Impact

SmoothQuant is widely adopted by industry

- NVIDIA FasterTransformer
- NVIDIA TRT-LLM
- MLPerf 8-bit: closed the accuracy gap.
- Intel Neural Compressor / Q8-Chat on Xeon
- Meta/Microsoft/Amazon/HuggingFace …

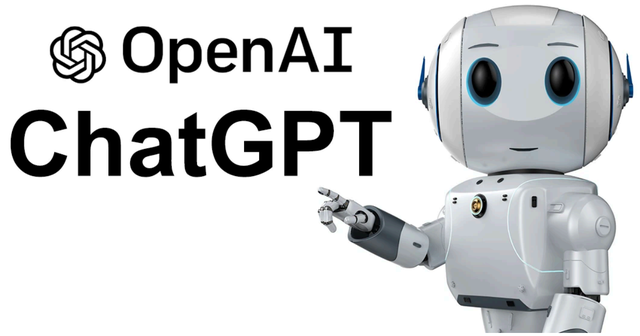
Efficient Streaming Language Models with Attention Sinks

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

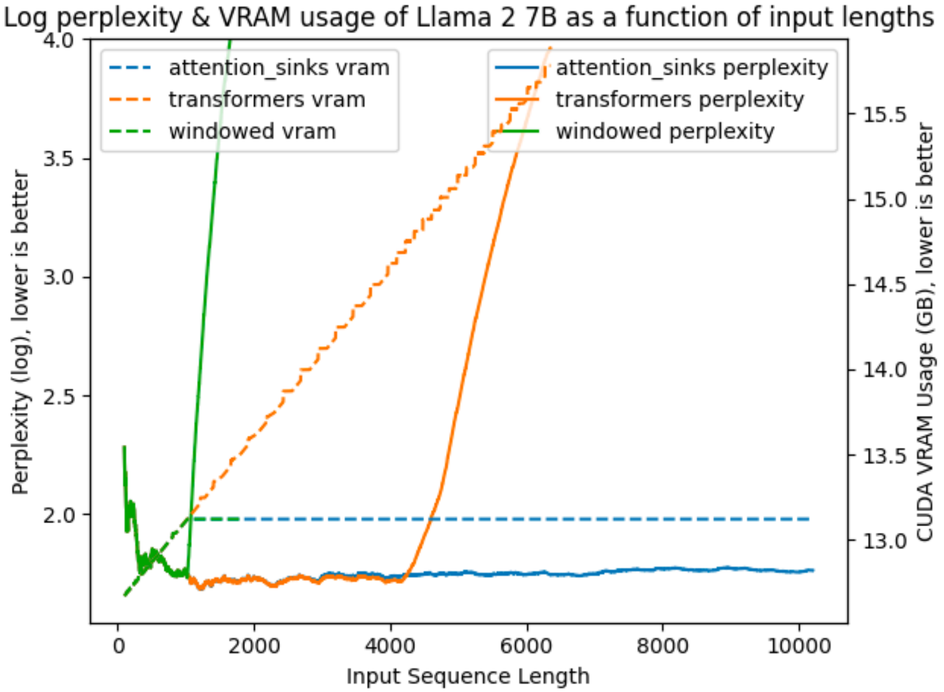
Massachusetts Institute of Technology[1]
Meta AI[2]
Carnegie Mellon University[3]
NVIDIA[4]

Challenges of Deploying LLMs in Streaming Applications

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

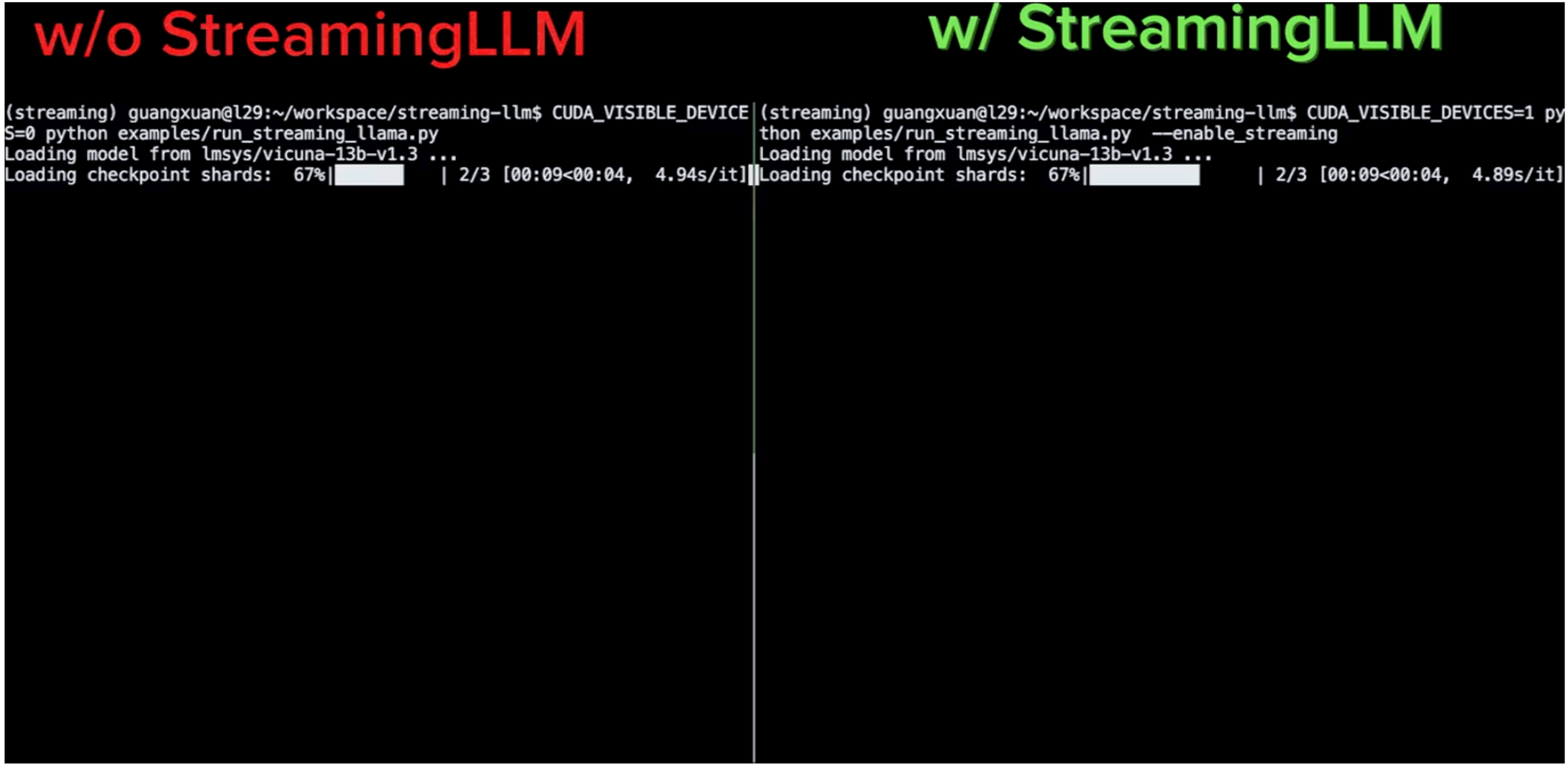


- Challenges:
  - Extensive memory consumption during the decoding stage.
  - Inability of popular LLMs to generalize to longer text sequences.

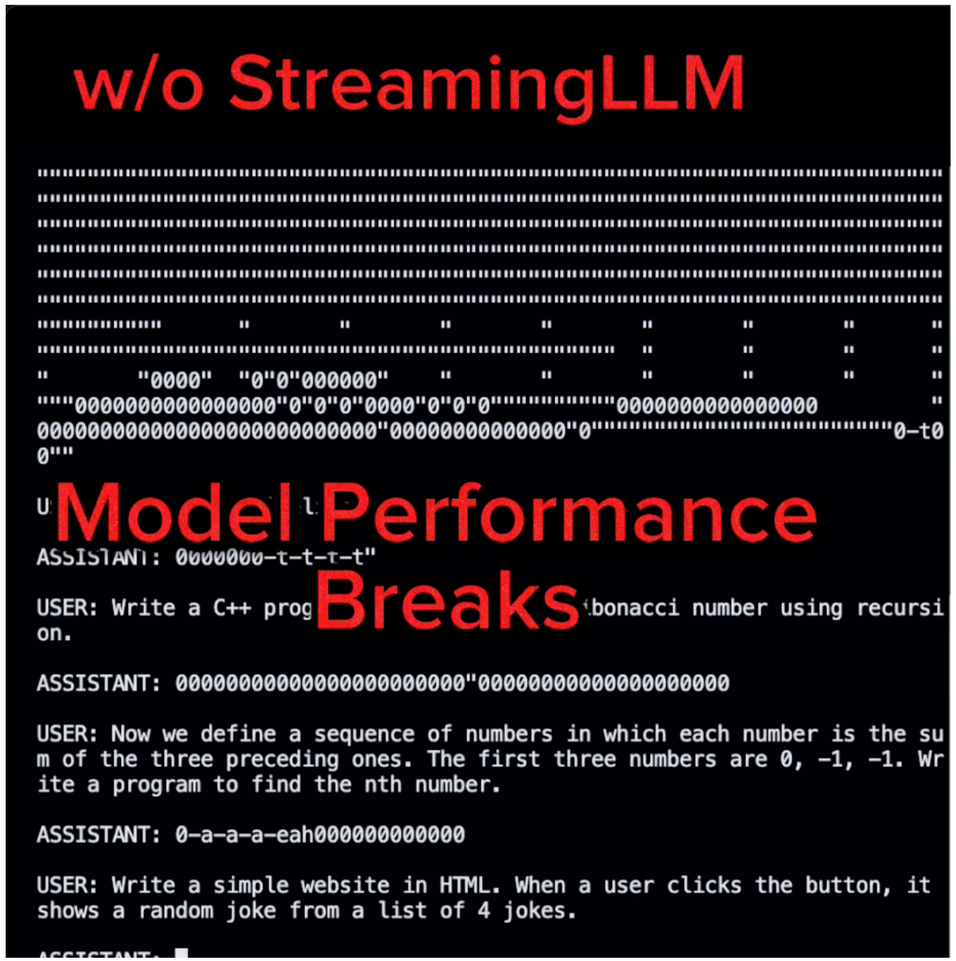


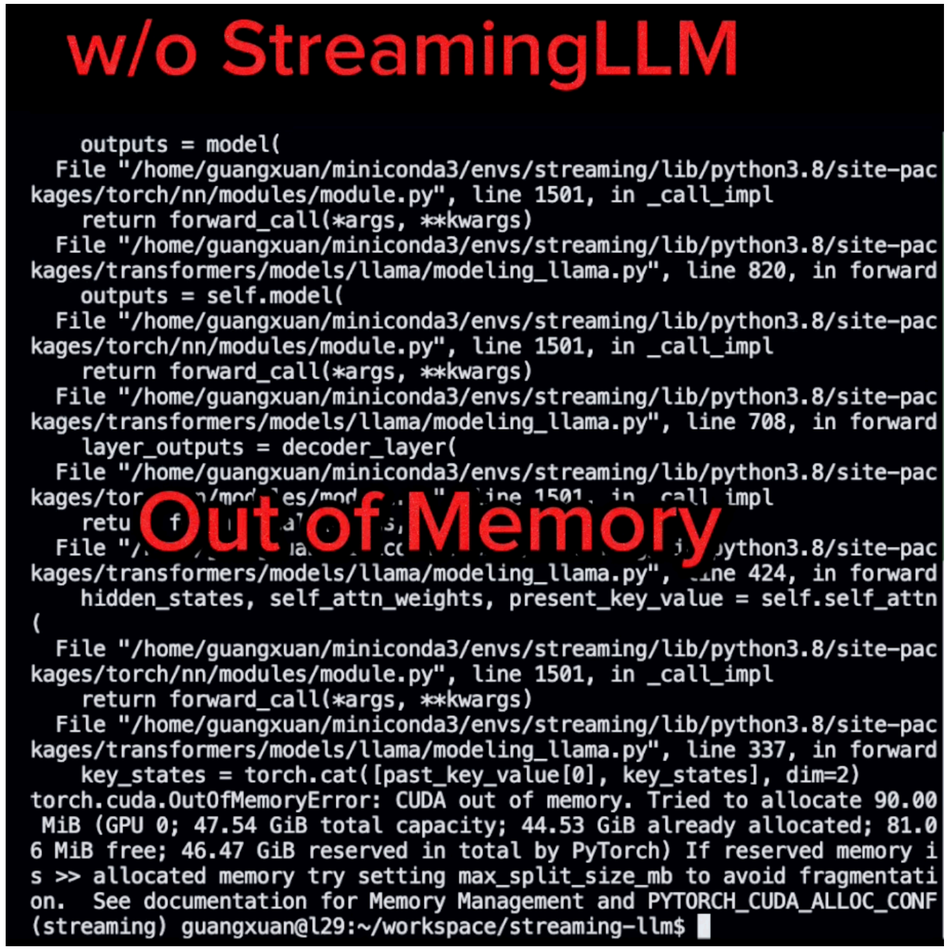
https://github.com/tomaarsen/attention_sinks

Challenges of Deploying LLMs in Streaming Applications



Challenges of Deploying LLMs in Streaming Applications

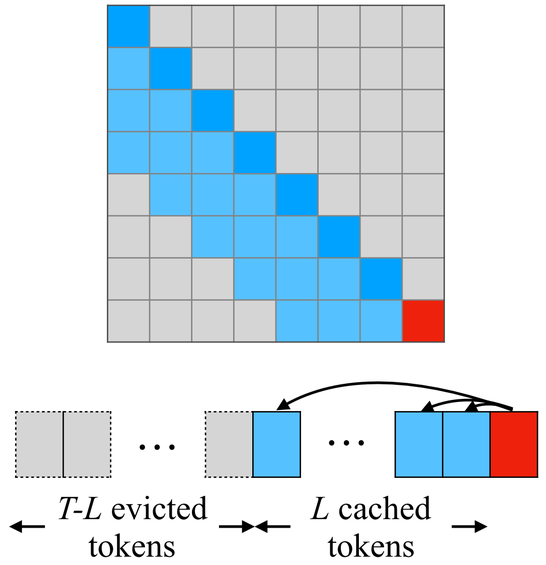




The Limits of Window Attention

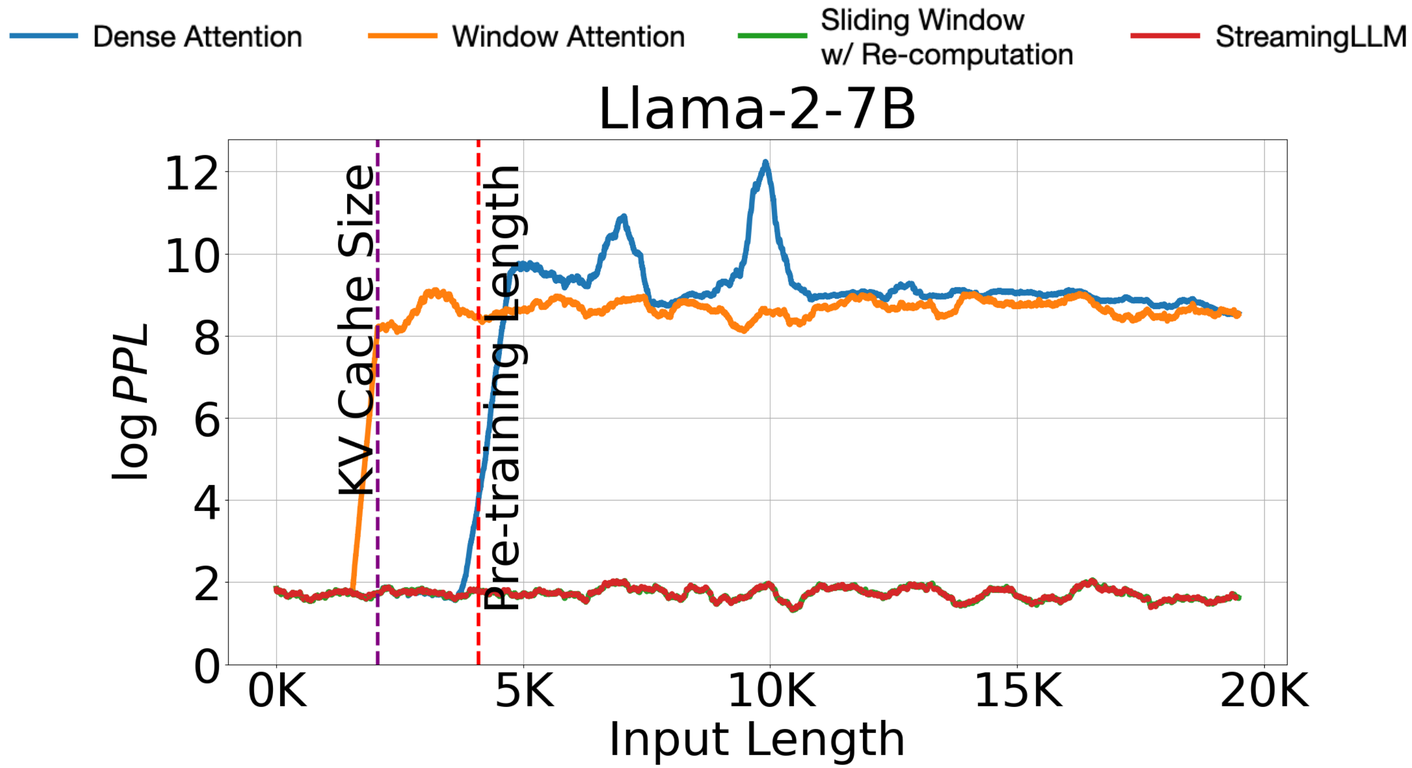
- A natural approach — window attention: caching only the most recent Key-Value states.
- Drawback: model collapses when the text length surpasses the cache size, when the initial token is evicted.

(b) Window Attention



$O(TL)$ ✔ **PPL:** 5158

Breaks when initial tokens are evicted.



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.

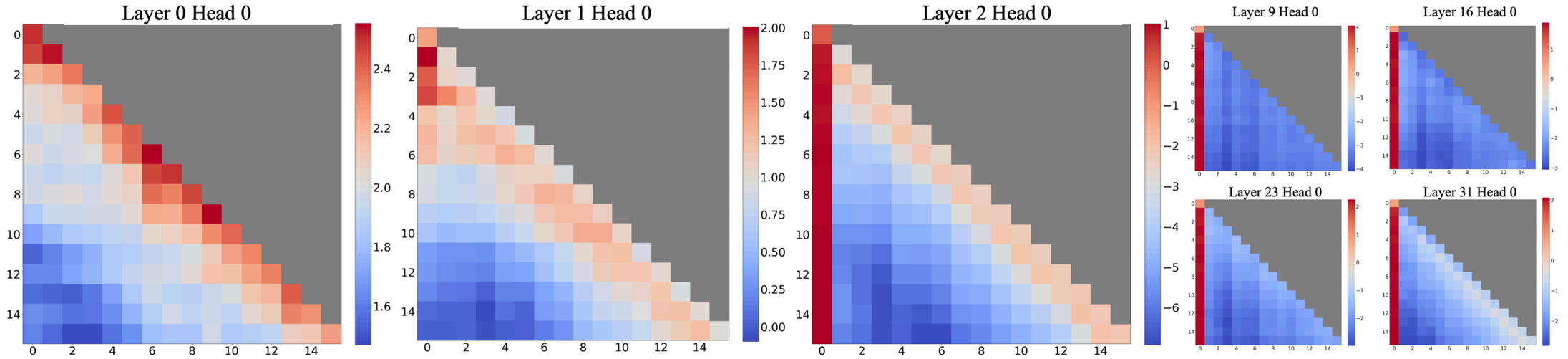


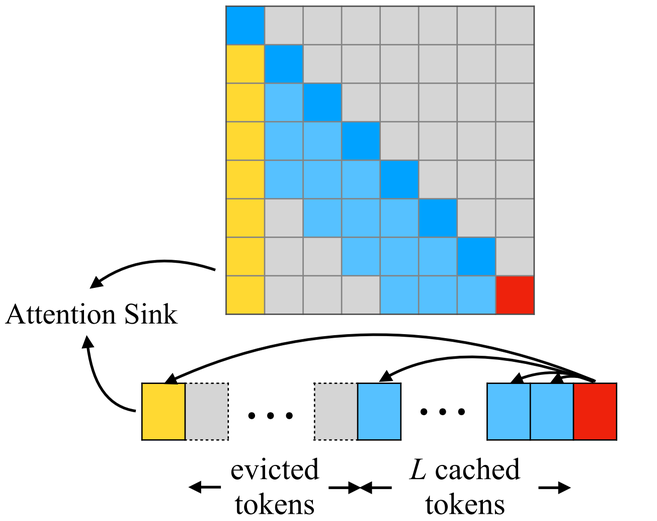
Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two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

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

StreamingLLM: Using Attention Sinks for Infinite Streams

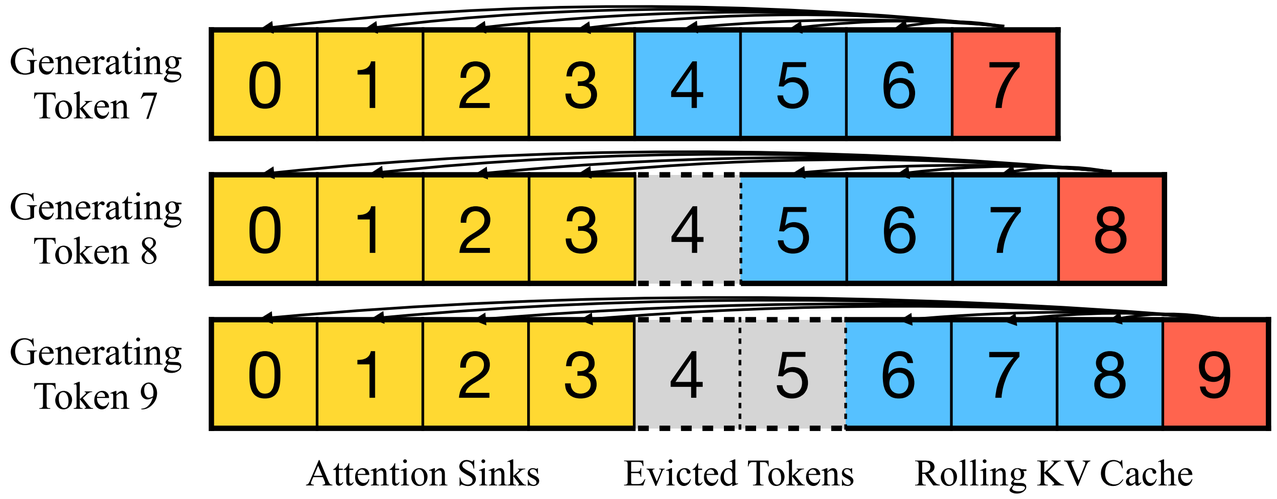
- **Objective:** Enable LLMs trained with a finite attention window to handle infinite text lengths without additional training.
- **Key Idea:** **preserve the KV of attention sink tokens**, along with the sliding window's KV to stabilize the model's behavior.

(d) **StreamingLLM (ours)**



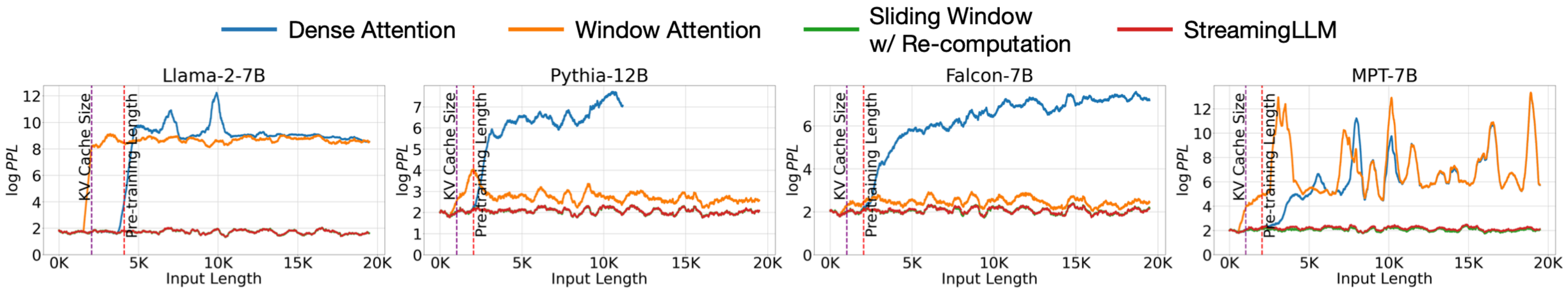
$O(TL)$ ✔ **PPL:** 5.40 ✔

Can perform efficient and stable language modeling on long texts.



Streaming Performance

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

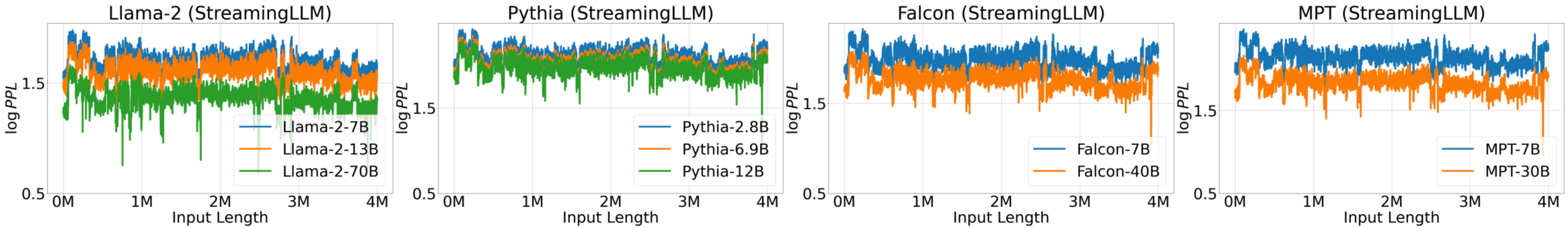


- Dense attention fails beyond pre-training attention window size.
- Window attention fails after input exceeds cache size (initial tokens evicted).
- StreamingLLM shows stable performance; perplexity close to sliding window with re-computation baseline.

Streaming Performance

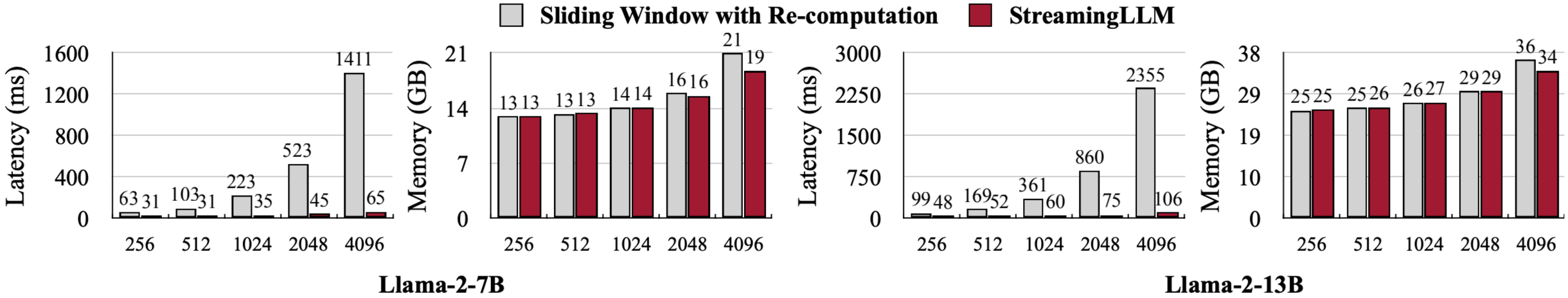
Super Long Language Modeling

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



Efficiency

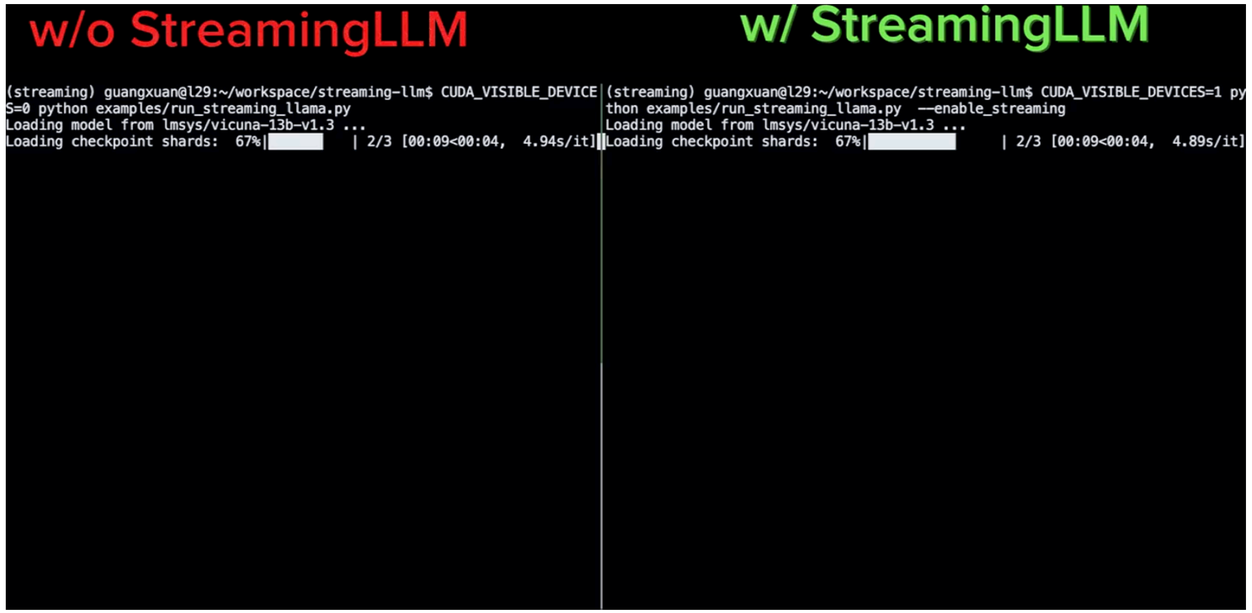
- **Comparison baseline:** The sliding window with re-computation, a method that is computationally heavy due to quadratic attention computation within its window.
- StreamingLLM provides up to 22.2x speedup over the baseline, making LLMs for real-time streaming applications feasible.



Conclusion

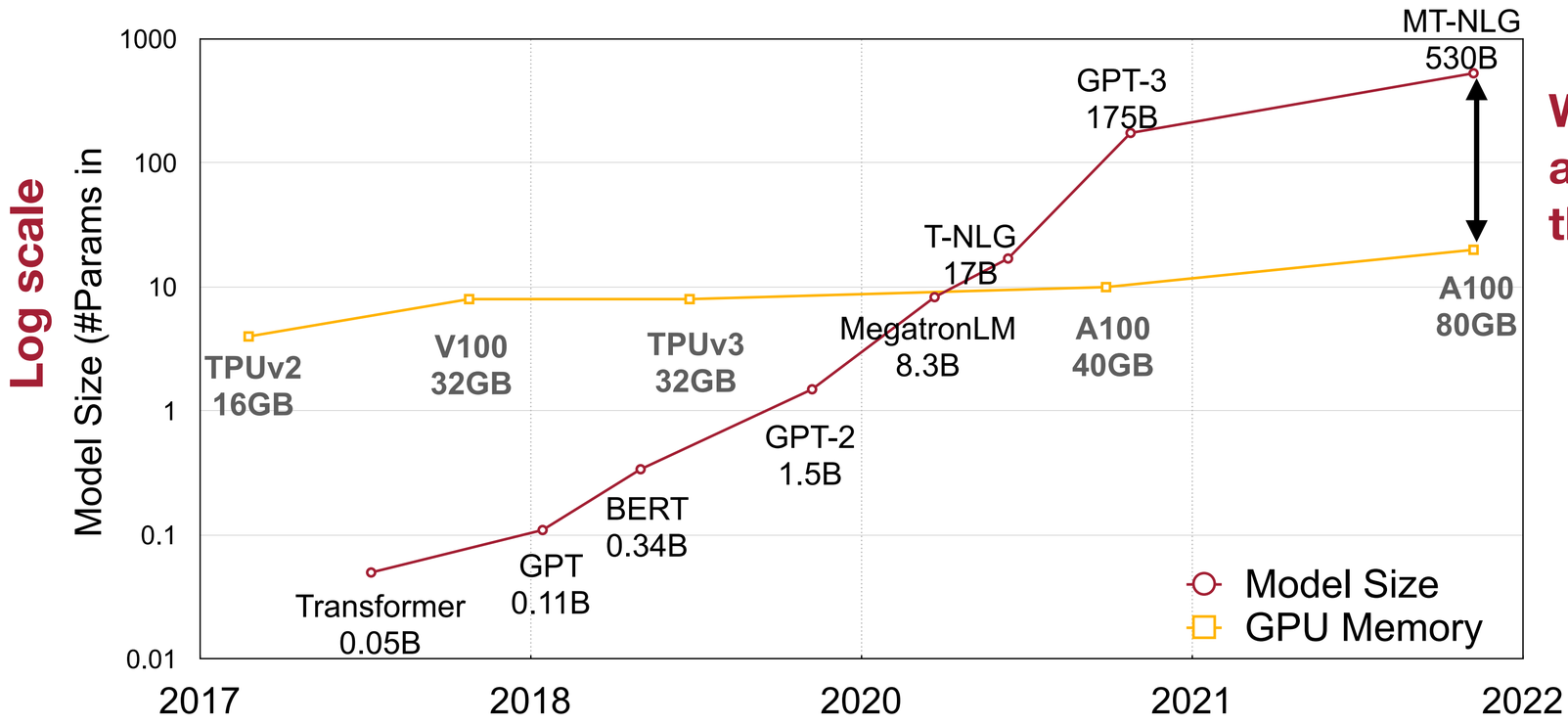
- We propose StreamingLLM, enabling the streaming deployment of LLMs.

- Paper: https://arxiv.org/abs/2309.17453
- Code: https://github.com/mit-han-lab/streaming-llm **6.7K Stars**
- Demo: https://youtu.be/UgDcZ3rvRPg
- Integration:
  - NVIDIA TensorRT-LLM
  - Intel Transformer Extension
  - Huggingface Transformers



Thanks for Listening!

•



We need efficient algorithms to bridge the gap.