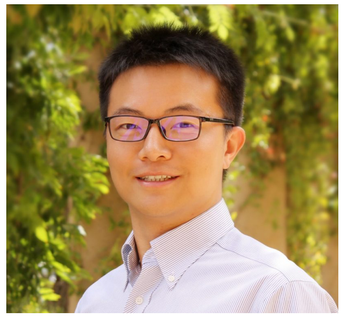
MIT HAN LAB

EfficientML.ai Lecture 15 Long-Context LLM



Song Han

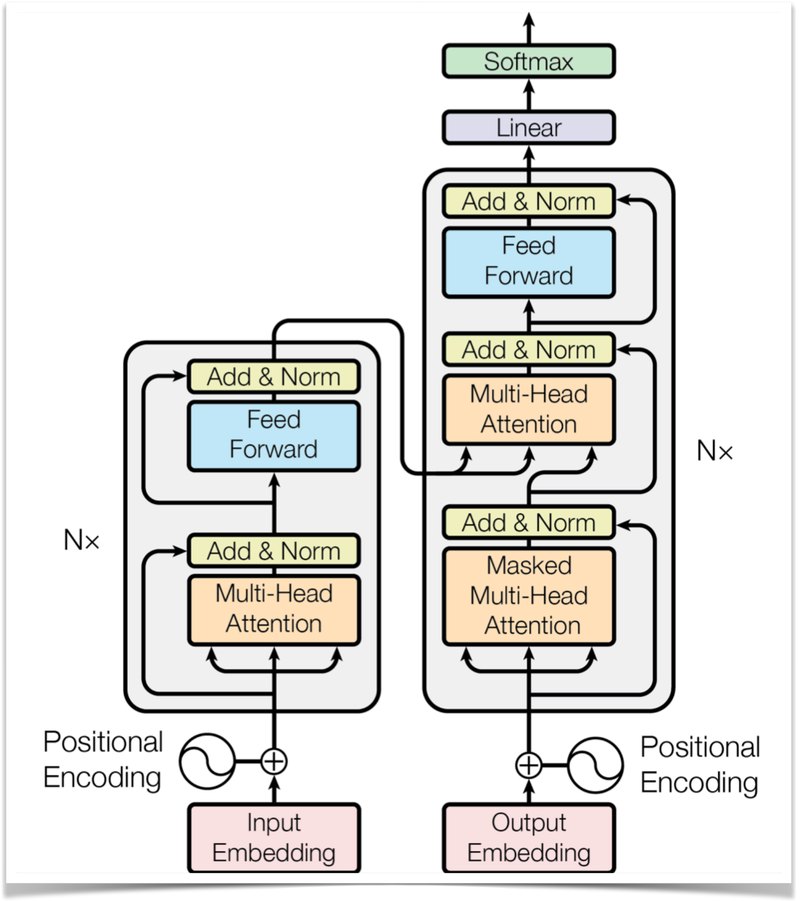
Associate Professor, MIT
Distinguished Scientist, NVIDIA

@SongHan_MIT

Lecture Plan

Today, we will cover:

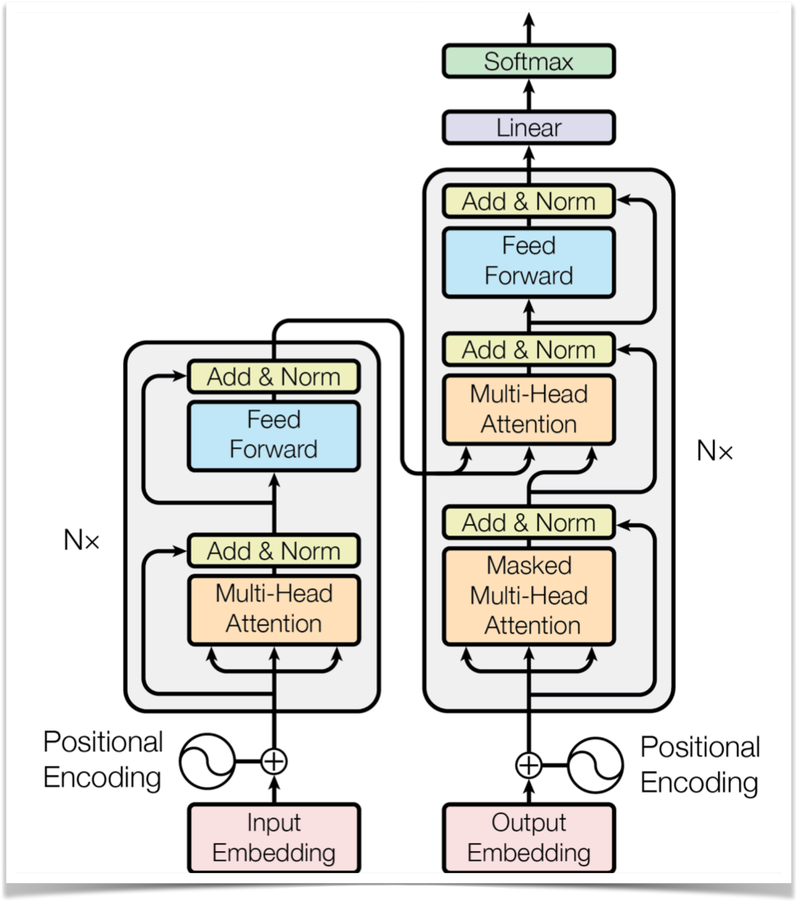
1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. **Beyond Transformers**
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



Lecture Plan

Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



Relative Positional Encoding

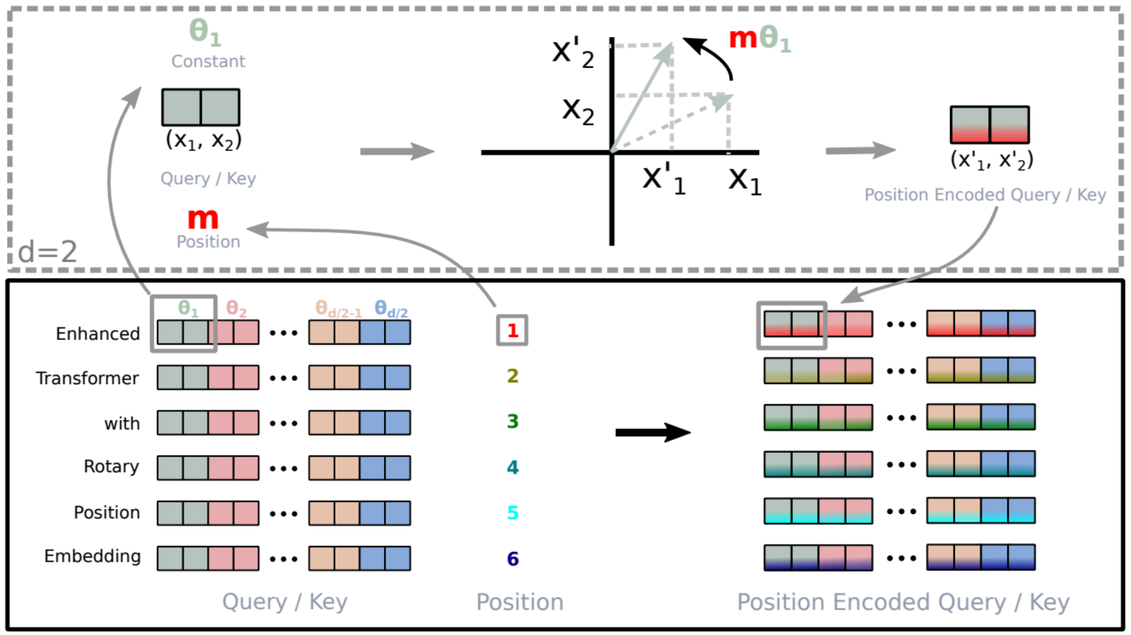
Rotary Positional Embedding (RoPE)

- A popular implementation for relative positional embedding (used in LLaMA)
- Rotate the embeddings in 2D space:
  - Split an embedding of dimension d into $d/2$ pairs, each pair considered as a 2D coordinate
  - Apply rotation according to the position m

We need a big enough number to distinguish more tokens

$$\Theta = \{\theta_i = \boxed{10000}^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$$

$$\begin{aligned}
\text{RoPE}(x, m) &= xe^{mi\varepsilon} \\
\langle \text{RoPE}(q_j, m), \text{RoPE}(k_j, n) \rangle &= \langle q_j e^{mi\varepsilon}, k_j e^{ni\varepsilon} \rangle \\
&= q_j k_j e^{mi\varepsilon} \overline{e^{ni\varepsilon}} \\
&= q_j k_j e^{(m-n)i\varepsilon} \\
&= \text{RoPE}(q_j k_j, m - n)
\end{aligned}$$



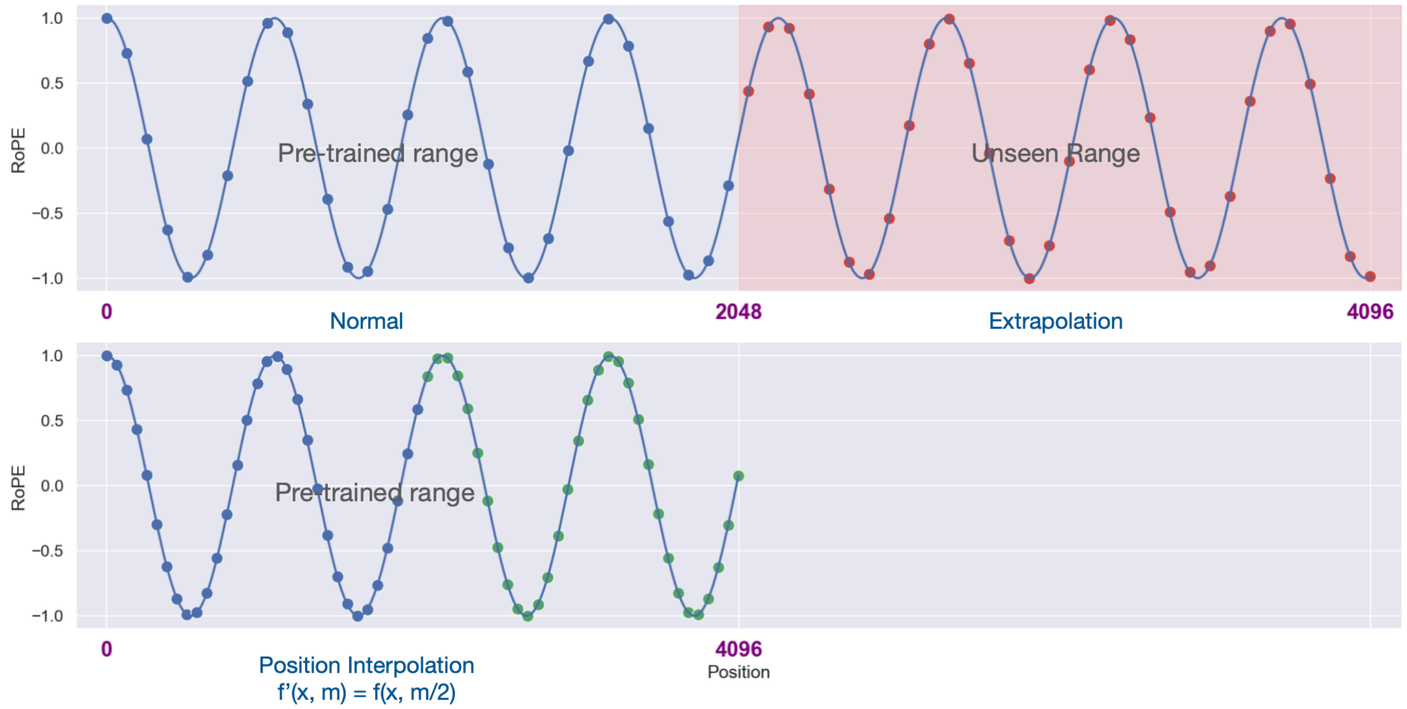
- The phase angle of the inner product of two complex vectors is the phase difference between the two complex vectors (thus m-n)

RoFormer: Enhanced Transformer with Rotary Position Embedding [Su *et al.*, 2021]

Relative Positional Embedding

Advantage of RoPE: extending the context window

- LLM is usually trained with a context length constraint (e.g., 2k for LLaMA, 4k for Llama-2, 8k for GPT-4) and fails with a larger context length
- We can extend the context length support by **interpolating** RoPE (i.e., use a smaller θ_i)
- Extend the context length of LLaMA from 2k to 32k



$m \in [0, 2048 * 2), \quad \theta_i' = \theta_i$ ✗

$m \in [0, 2048 * 2), \quad \theta_i' = \theta_i / 2$ ✓

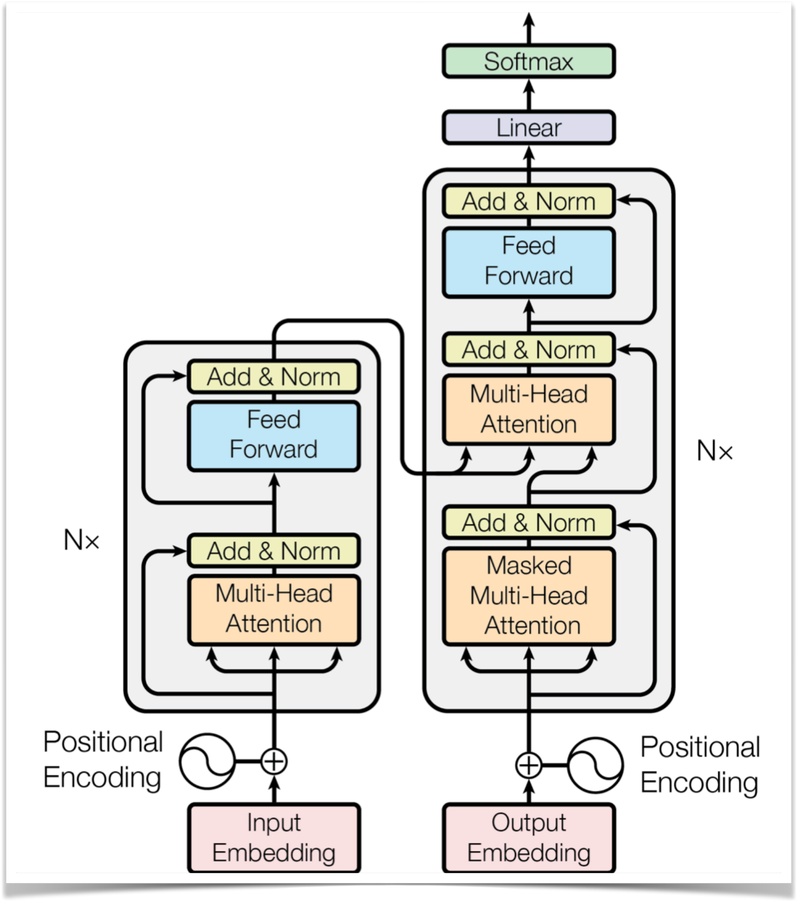
LLMs usually need to be fine-tuned after extending the context length!

Extending Context Window of Large Language Models via Position Interpolation [Chen et al., 2023]

Lecture Plan

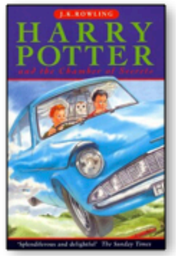
Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

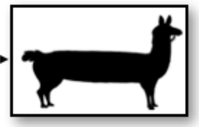
Below is a section in the book, **Harry Potter and the Chamber of Secrets**. {***book_content***} Now the paper ends. Please tell me that what high-level idea the author want to indicate in this book. *~ 30k tokens*



LLAMA 2

Warning: model does not support context sizes greater than 4096 tokens; expect poor results. (output: \n). ***Wrong Output***

Full fine-tune



128 TPUv3 for fine-tuning LLaMA to LongLLaMA [1]. ***Unaffordable training cost***

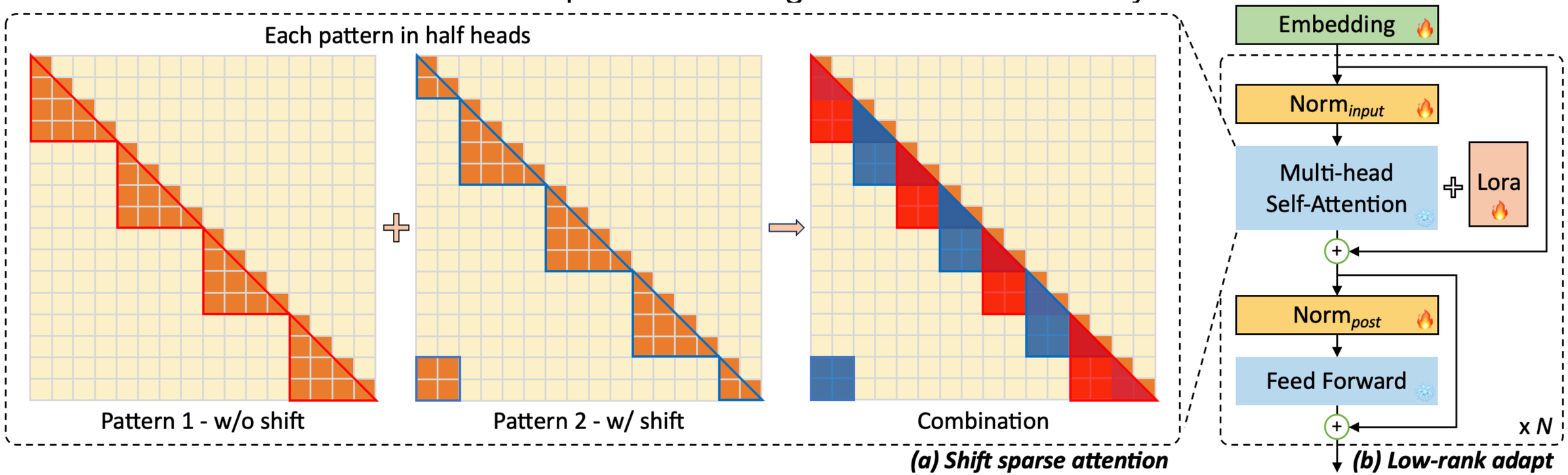
Long Alpaca models

LongLoRA [2] – Efficient fine-tuning LLMs to be long-context, e.g., 70B Llama2 to 32k on 8x A100 GPUs.

[1] Focused Transformer: Contrastive Training for Context Scaling. [Tworkowski, el al, NeurIPS 2023]
[2] LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

- **Attention is the bottleneck under long context**
- **Shifted Sparse Attention:**
  - Split attention heads, group tokens, shift the groups.
- **Enhanced LoRA:**
  - We should also fine-tune the input embedding and normalization layer.



LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

- **Shifted sparse attention @training**
  - Step1: Split attention heads.
  - Step2: Shift tokens by half group.
  - Step3: Reshape / group.
- **Full attention @inference.**

Algorithm 1: Pseudocode of Shift Sparse Attention in PyTorch-like style.

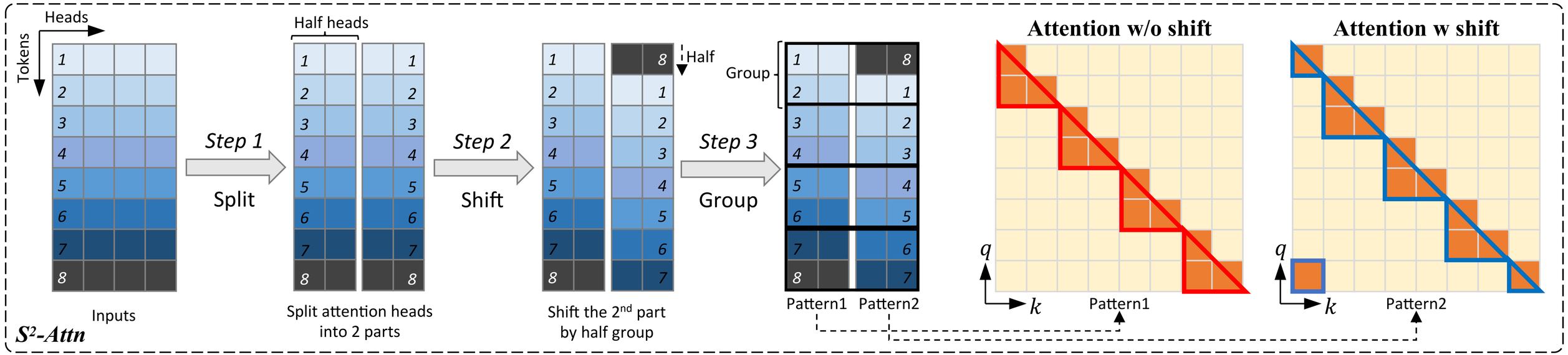
```
# B: batch size; S: sequence length or number of tokens; G: group size;
# H: number of attention heads; D: dimension of each attention head

# qkv in shape (B, N, 3, H, D), projected queries, keys, and values
# Key line 1: split qkv on H into 2 chunks, and shift G/2 on N
qkv = cat((qkv.chunk(2, 3)[0], qkv.chunk(2, 3)[1].roll(-G/2, 1)), 3).view(B*N/G,G,3,H,D)

# standard self-attention function
out = self_attn(qkv)

# out in shape (B, N, H, D)
# Key line 2: split out on H into 2 chunks, and then roll back G/2 on N
out = cat((out.chunk(2, 2)[0], out.chunk(2, 2)[1].roll(G/2, 1)), 2)
```

`cat`: concatenation; `chunk`: split into the specified number of chunks; `roll`: roll the tensor along the given dimension.



LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

- **Shifted sparse attention @training**
  - Saving training hours upon FlashAttn-2.
  - Better than other types of efficient attns.
- **Full attention @inference**

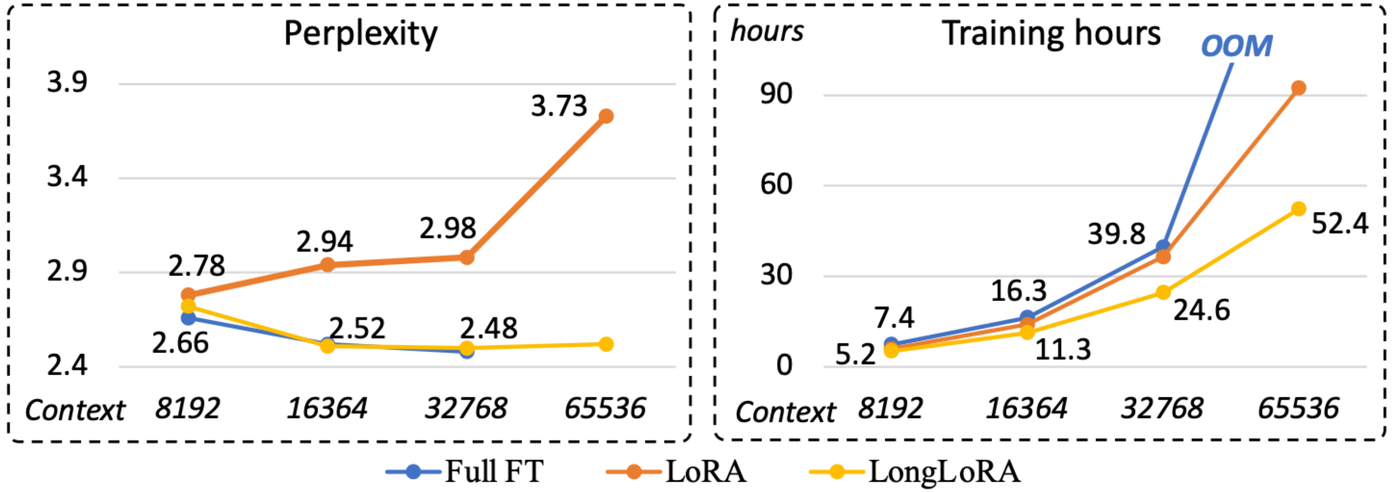


Table 2: **Comparisons among S^2-Attn and alternative attention patterns during fine-tuning**. We adapt a Llama2 7B model to 32768 context length with different attention patterns and improved LoRA at training time. We include four typical efficient attention designs, *e.g.*, shift, dilate (Ding et al., 2023), block sparse (Qiu et al., 2020), stride sparse (Child et al., 2019) for comparison. '*cro. heads / layers*' means to swap different attention settings across attention *heads* or sequential *layers*. Taking S^2-Attn as an example, '*cro. layers*' is to swap between w/ and w/o shift in sequential self-attention layers. '*only P1/P2*' means all attention heads use pattern 1 (all no shift) or Pattern 2 (all shift) in Figure 2. We visualize the patterns of different attention in the appendix.

Test w/ Full-Attn	S^2-Attn **cro. heads**	S^2-Attn *cro. layers*	S^2-Attn *only P1.*	S^2-Attn *only P2.*	Dilate *cro. heads*	Block sparse *cro. heads*	Stride sparse *cro. heads*
✗	8.64	8.63	9.17	9.64	8.75	11.49	32.81
✓	8.12	9.70	8.39	9.81	11.78	8.30	24.03

LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

- **Enhance LoRA by:**
  - + Input embedding.
  - + Normalization layers.

✓ Limited additional cost.

  - Norm < 0.004%, embeddings < 2%

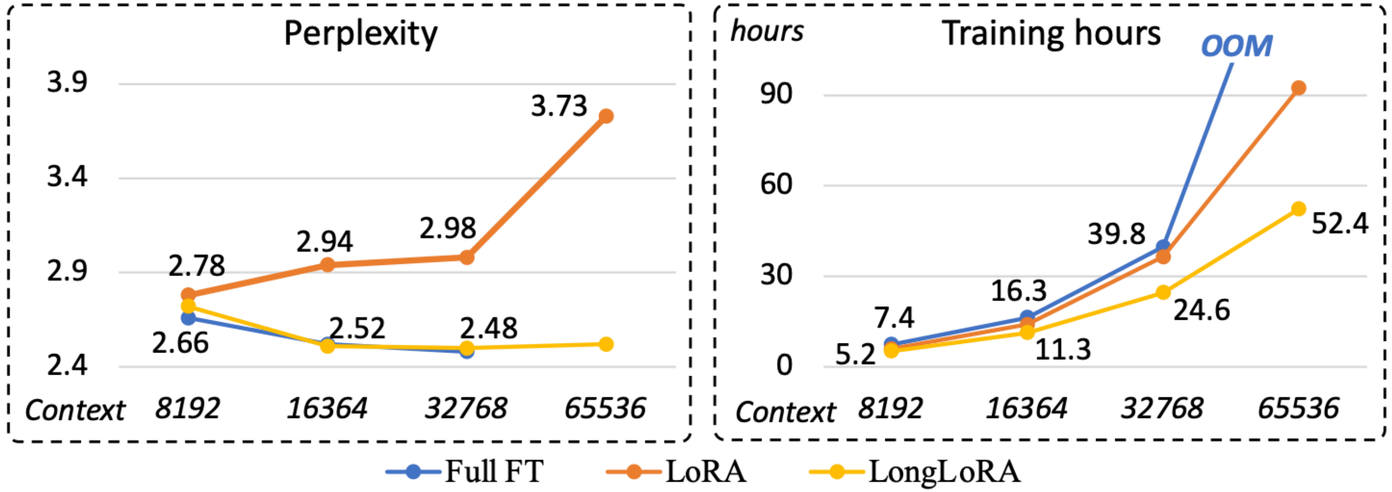


Table 3: **Finetuning normalization and embedding layers is crucial for low-rank long-context adaptation**. Llama2 7B (Touvron et al., 2023b) models with the proposed S^2-Attn are trained on the RedPajama (Computer, 2023) dataset. The target context length is 32768. '+ Normal / Embed' means normalization or embedding layers are trainable. Perplexity results are evaluated on PG19 (Rae et al., 2020) validation set. For long context adaptation, there is a large performance gap between standard LoRA (Hu et al., 2022) and full fine-tuning. Without trainable normalization or embeddings, larger ranks in LoRA can not close this gap.

Method	Full FT	LoRA (rank)						LoRA (rank = 8)	
		8	16	32	64	128	256	+ Norm	+ Norm + Embed
PPL	8.08	11.44	11.82	11.92	11.96	11.97	11.98	10.49	8.12

LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

LongLoRA: Efficient Fine-Tuning of Long-Context LLMs

- **[Evaluation] Retrieval**
  - Topic retrieval on LongChat.
  - Passkey retrieval.

✓ Feasible to the fine-tuned context.

```
There is an important info hidden inside a lot of irrelevant text.
Find it and memorize them.  I will quiz you about the important
information there.
The grass is green.  The sky is blue.  The sun is yellow.  Here we
go.  There and back again.  (repeat M times)
The pass key is 12362.  Remember it.  12362 is the pass key.
The grass is green.  The sky is blue.  The sun is yellow.  Here we
go.  There and back again.  (repeat N times)
What is the pass key?  The pass key is
```

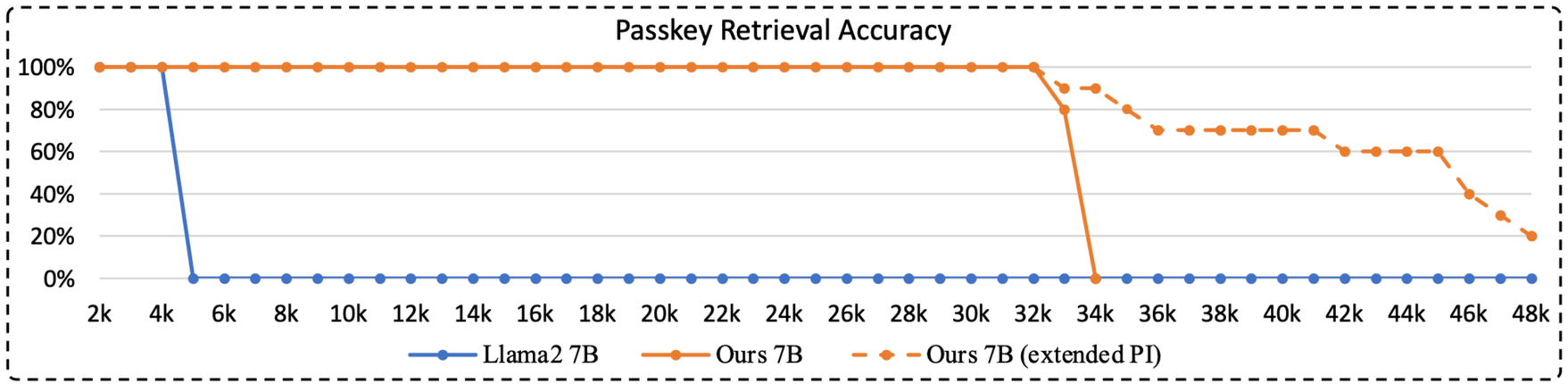


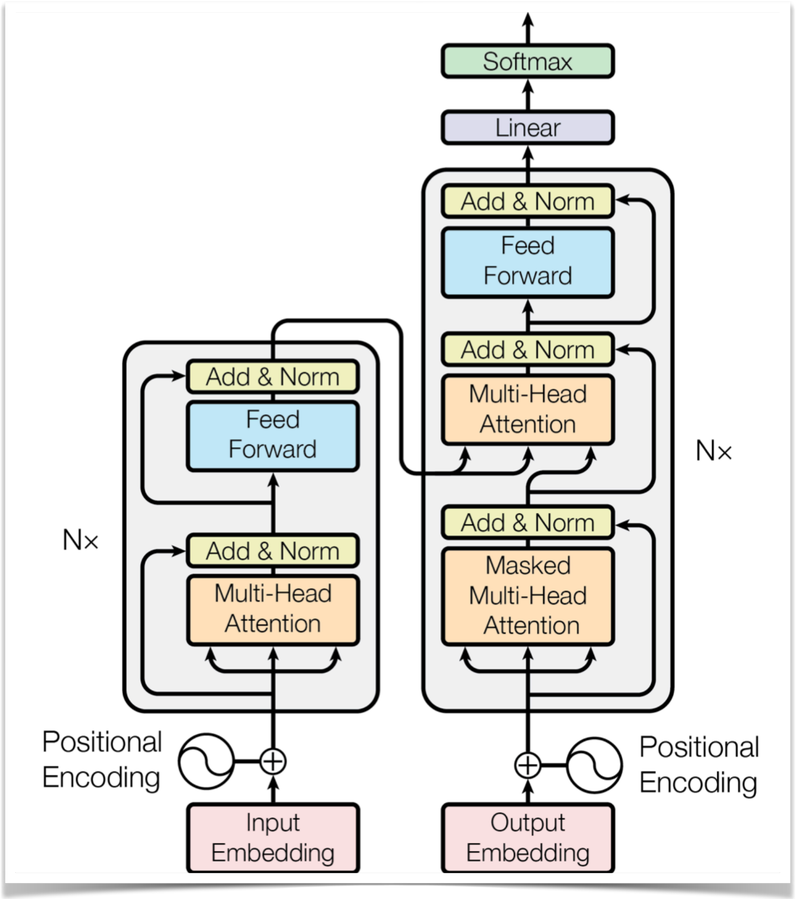
Figure 4: Accuracy comparison on passkey retrieval between Llama2 7B and our 7B model fine-tuned on 32768 context length. Our model presents no retrieval accuracy degradation until 33k or 34k, which exceeds the context length. It can further enhance its capability of long sequence modeling through a straightforward extension of position embeddings, without additional fine-tuning.

LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. [Chen et al, Arxiv 2023]

Lecture Plan

Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba

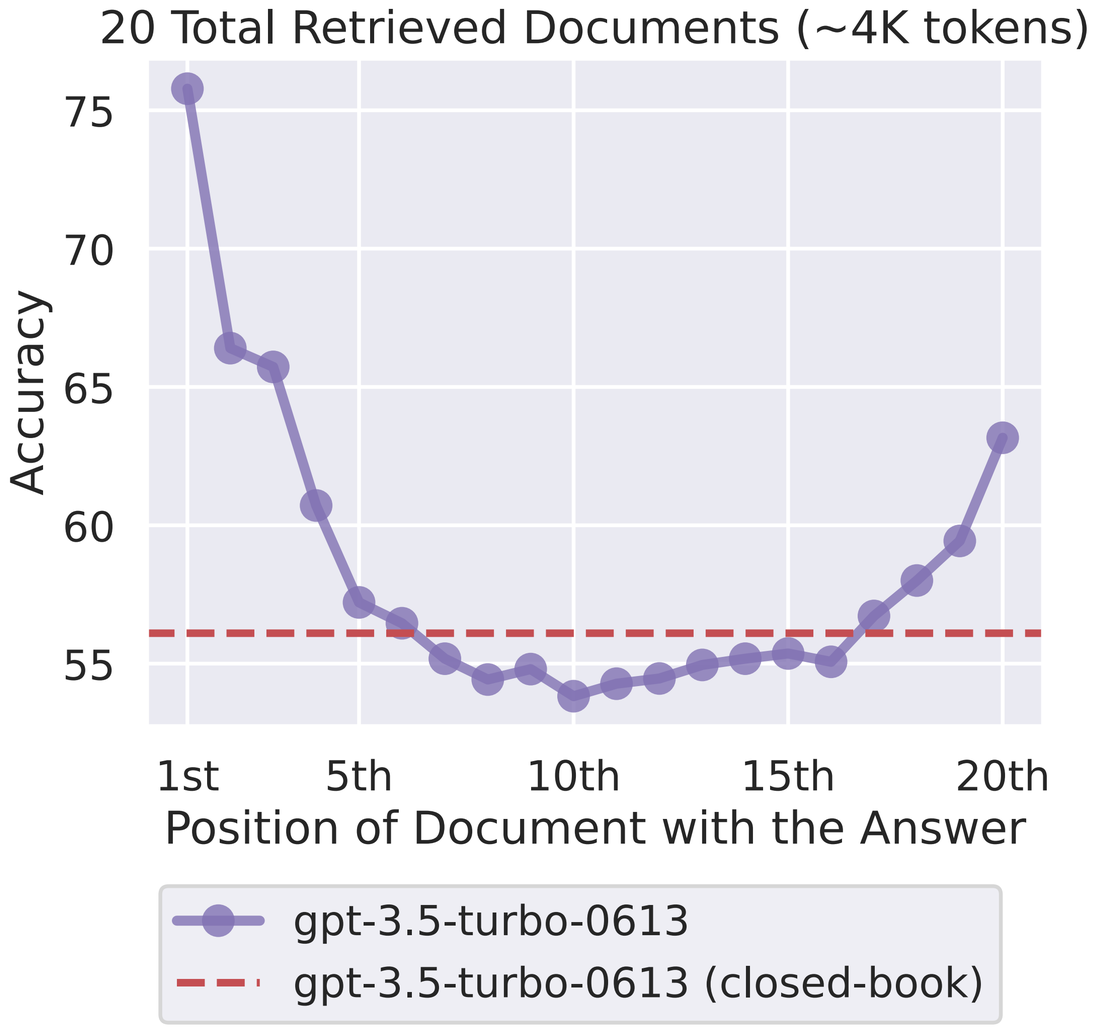


The Lost in the Middle Phenomenon

How Do Language Models Use Long Contexts?

- **Challenge:** Modern language models process large text contexts but struggle to leverage information effectively across long contexts.
- **"Lost in the Middle":** Models show a **U-shaped performance curve:**
  - **High accuracy** when relevant info appears at the **start or end** of the context.
  - **Low accuracy** when relevant info is in the **middle.**

- **Key Insight**: Fluent long-context responses don't imply effective long-context processing.
- **Takeaway:** Rigorous evaluation is essential to understand and improve context utilization in LLMs.

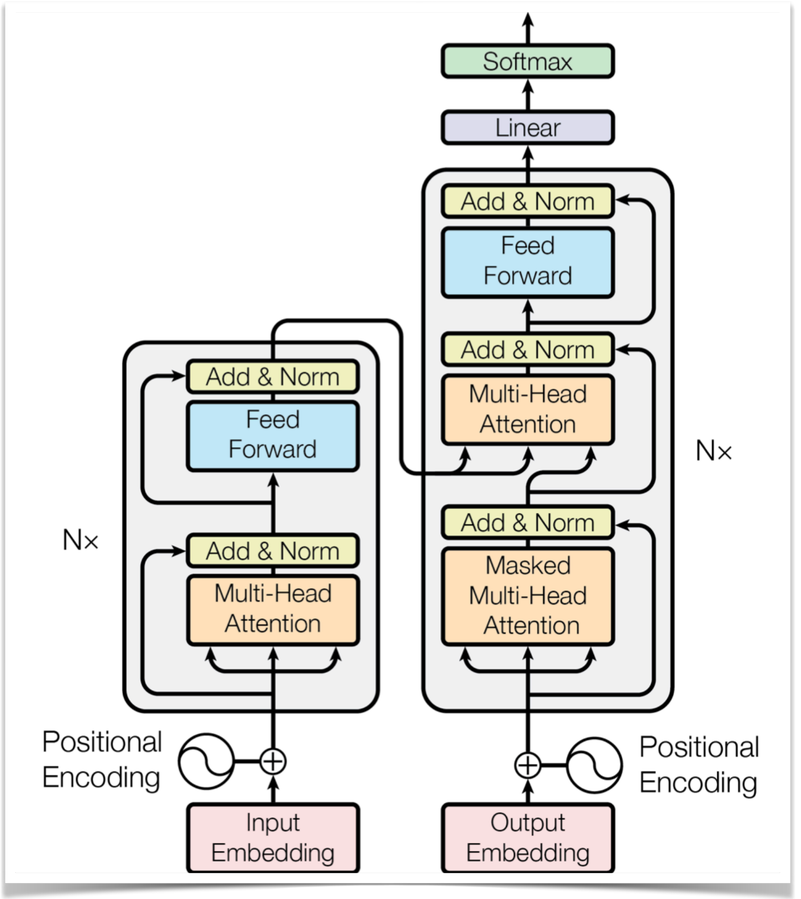


Lost in the Middle: How Language Models Use Long Contexts [Liu et al., 2023]

Lecture Plan

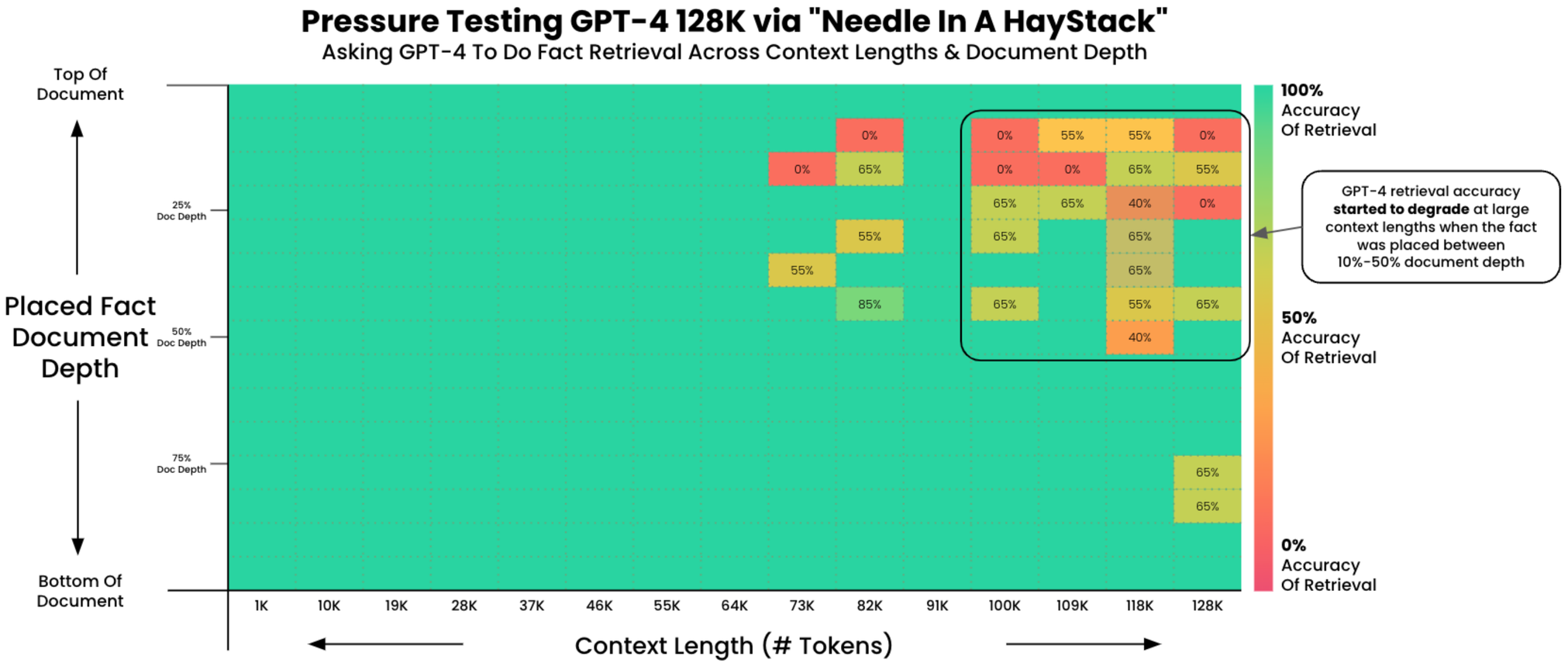
Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



Needle In A Haystack

How Do Language Models Perform under Long Contexts?



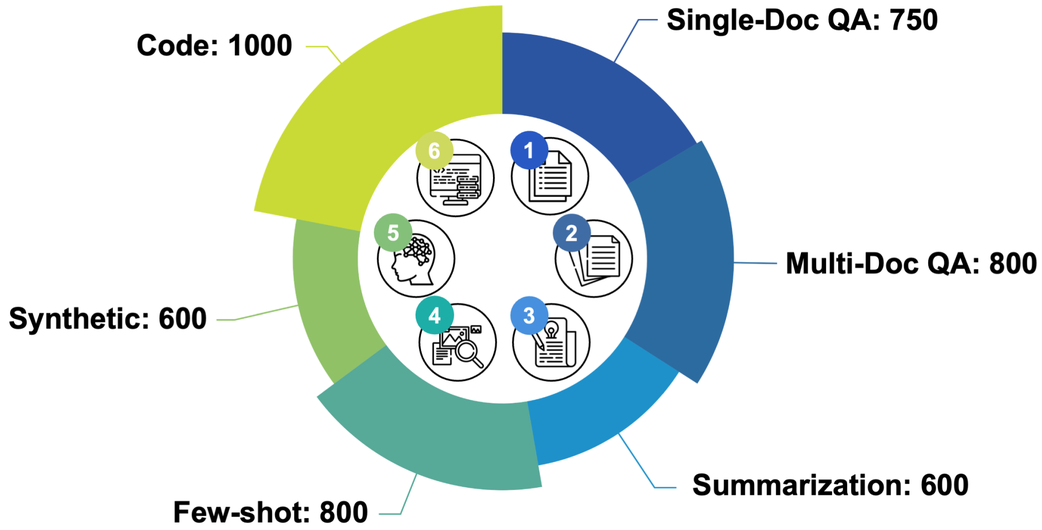
- **Inserting information (Needle) at different depth of a long document (Haystack):** "The best thing in San Francisco is eating a sandwich and sitting in Dolores Park on a sunny day."
- **Ask the model at the end of the document (Haystack) to recall the information (Needle):** "What is the best thing to do in San Francisco?"

https://github.com/gkamradt/LLMTest_NeedleInAHaystack

LongBench

Evaluating LLMs' Long Context Abilities More Comprehensively

- **Problem:** Solely evaluating on synthetic tasks has limited real-world relevance.
- **LongBench Benchmark:**
  - Includes 21 datasets across 6 task types (QA, summarization, few-shot learning, etc.) in English and Chinese.
  - Supports contexts up to 13,000+ tokens, with automated evaluations using metrics like F1 and ROUGE.
- **Key Findings:**
  - Techniques like scaled position embeddings improve long-context understanding.
  - Retrieval and context compression help but aren't as effective as models inherently designed for long contexts.

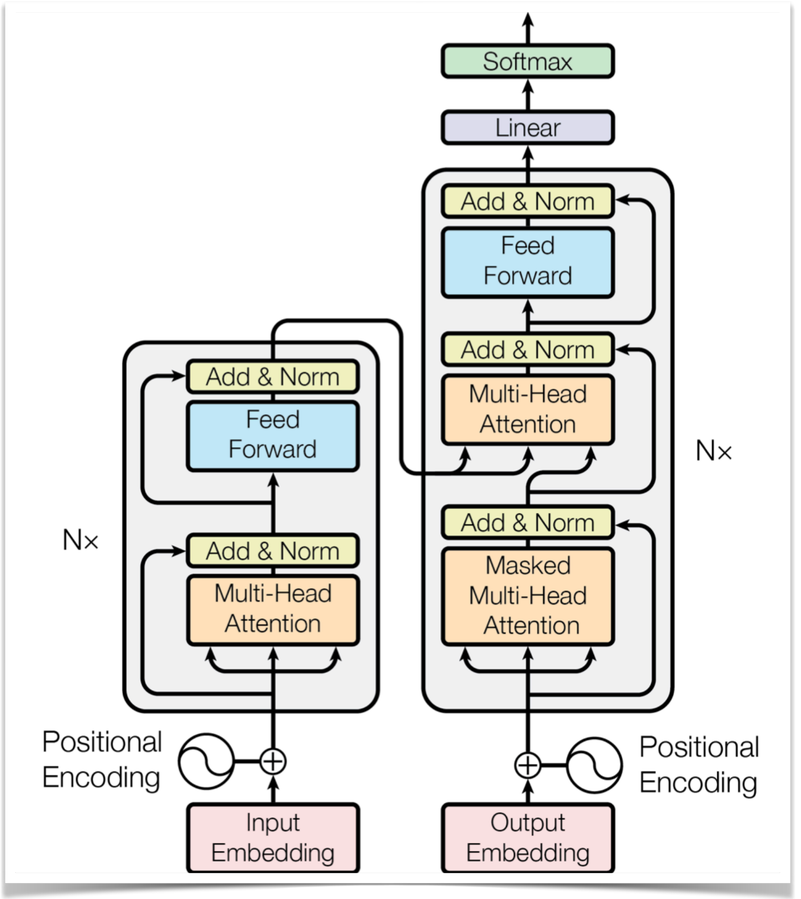


LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding [Bai et al., ACL 2024]

Lecture Plan

Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



KV Cache Optimizations

The KV cache could be large with long context

- We can calculate the memory required to store the KV cache
- Consider Llama-2-70B (if using MHA), KV cache requires

$$\underbrace{BS}_{minibatch} * \underbrace{80}_{layers} * \underbrace{64}_{kv-heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2\text{bytes}}_{FP16} = 2.5\text{MB} \times BS \times N$$

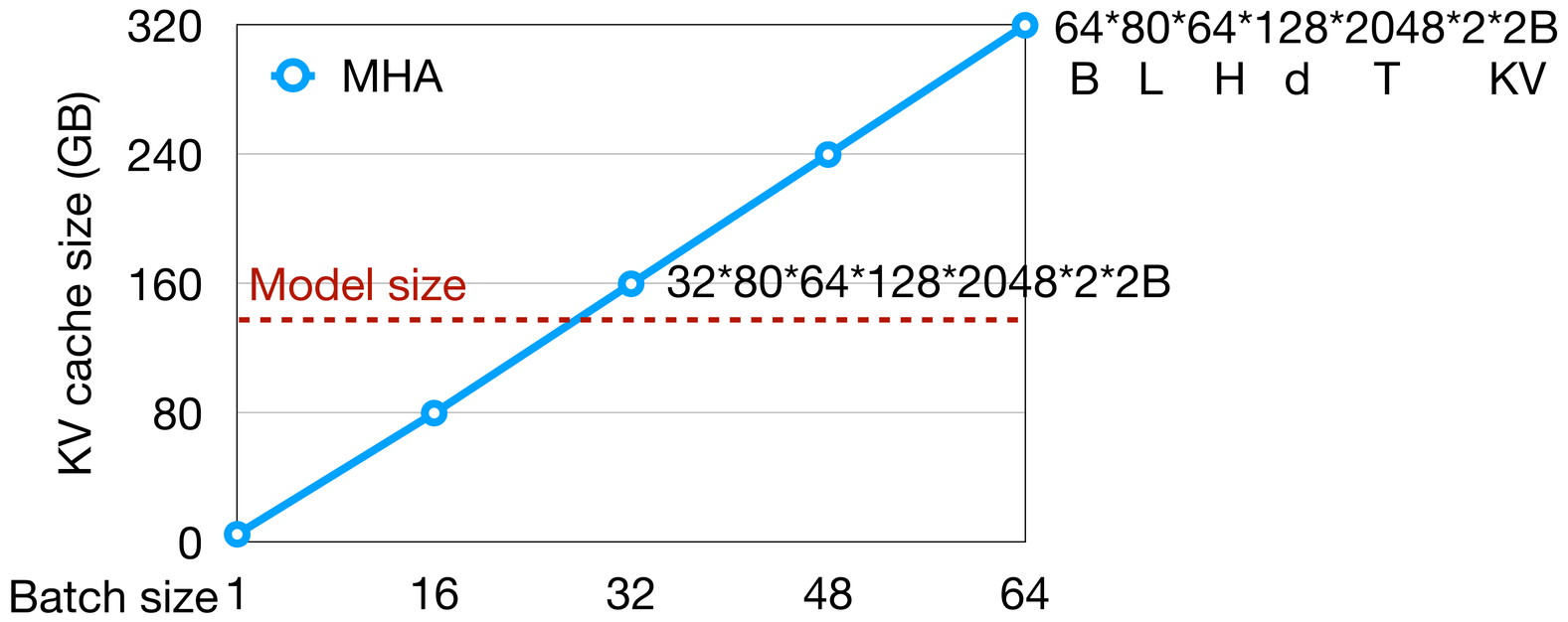
- bs=1, n_seq=512: 1.25GB
- bs=1, n_seq=4096 : 10GB (~ a paper)
- bs=16, n_seq=4096: 160GB (requires two A100 GPUs!)

https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices

KV Cache Optimizations

The KV cache could be large with long context

- Now, we calculate the KV cache size under $N = 2048$ different batch sizes
  - The KV cache size goes quickly larger than the model weights

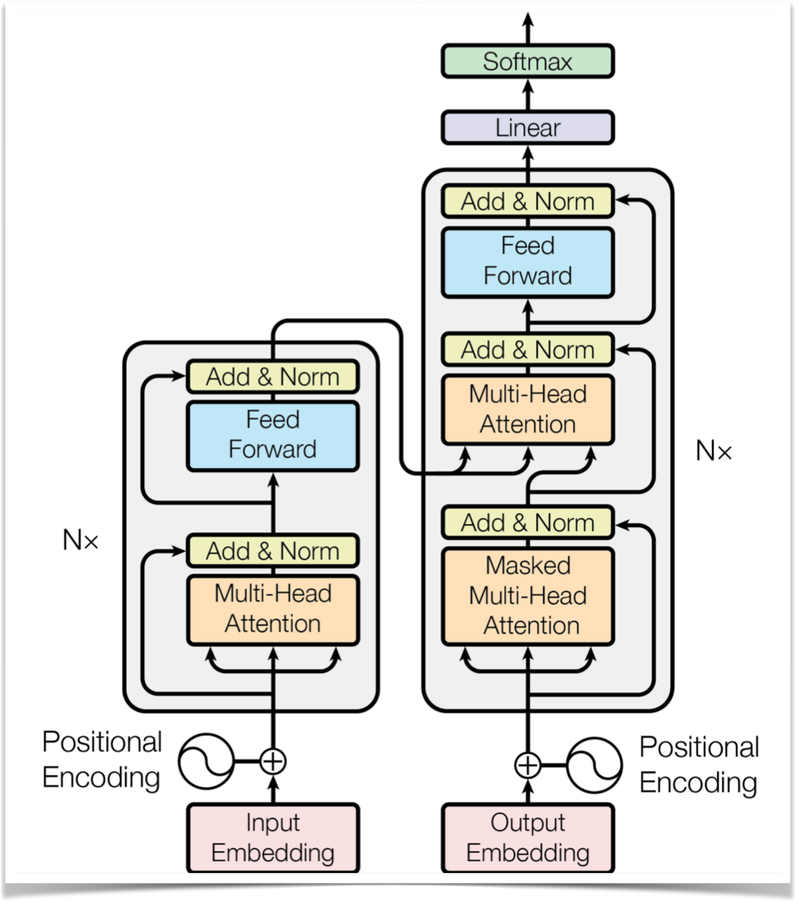


https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices

Lecture Plan

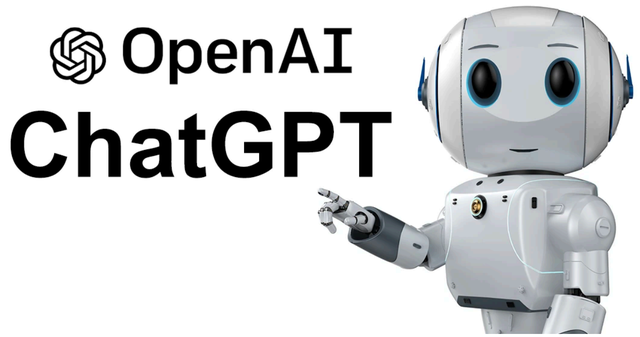
Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba

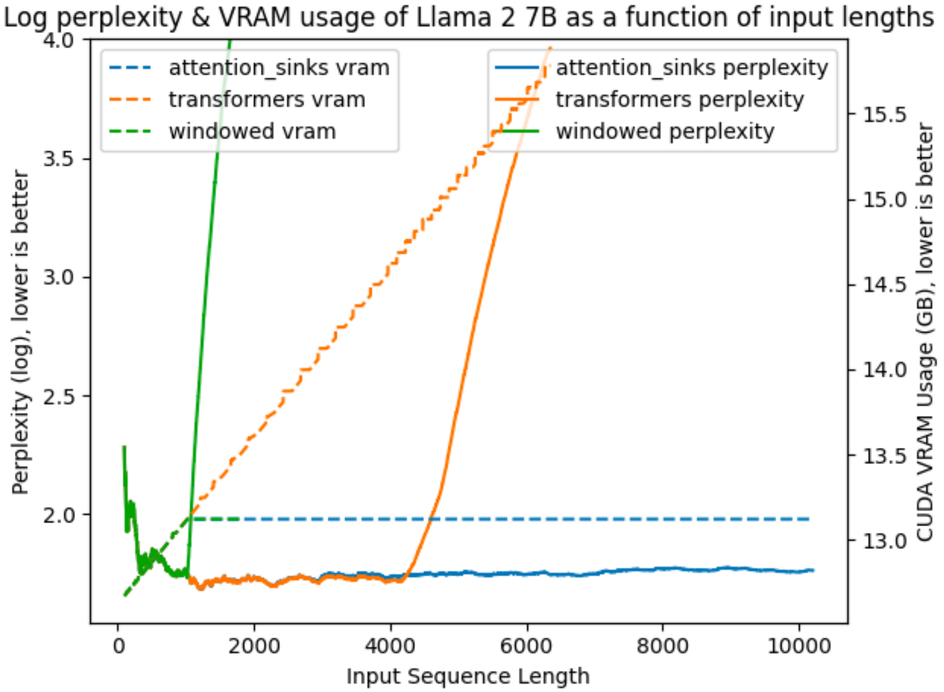


Challenges of Deploying LLMs in Streaming Applications

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

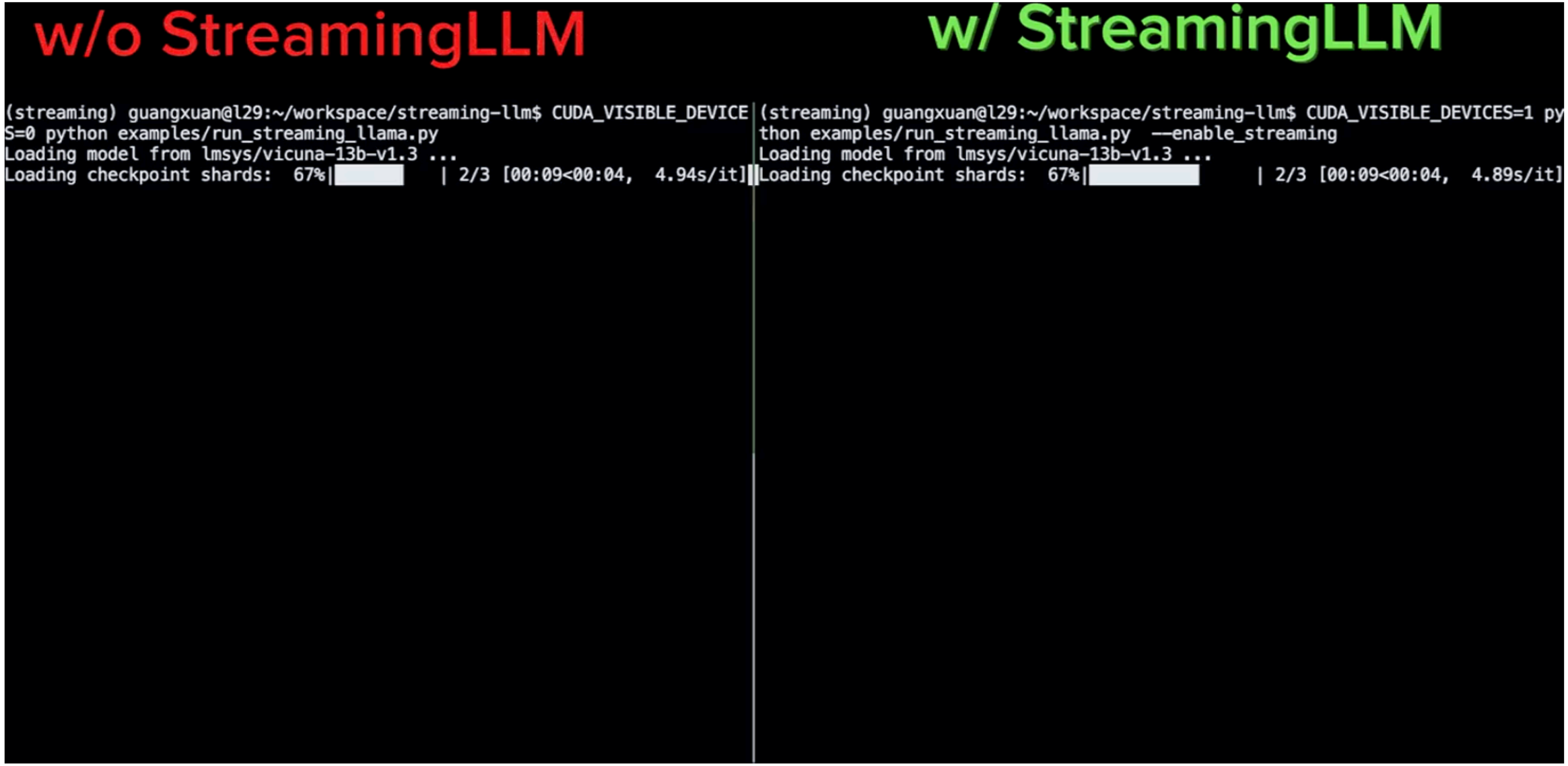


- Challenges:
  - Extensive memory consumption during the decoding stage.
  - Inability of popular LLMs to generalize to longer text sequences.

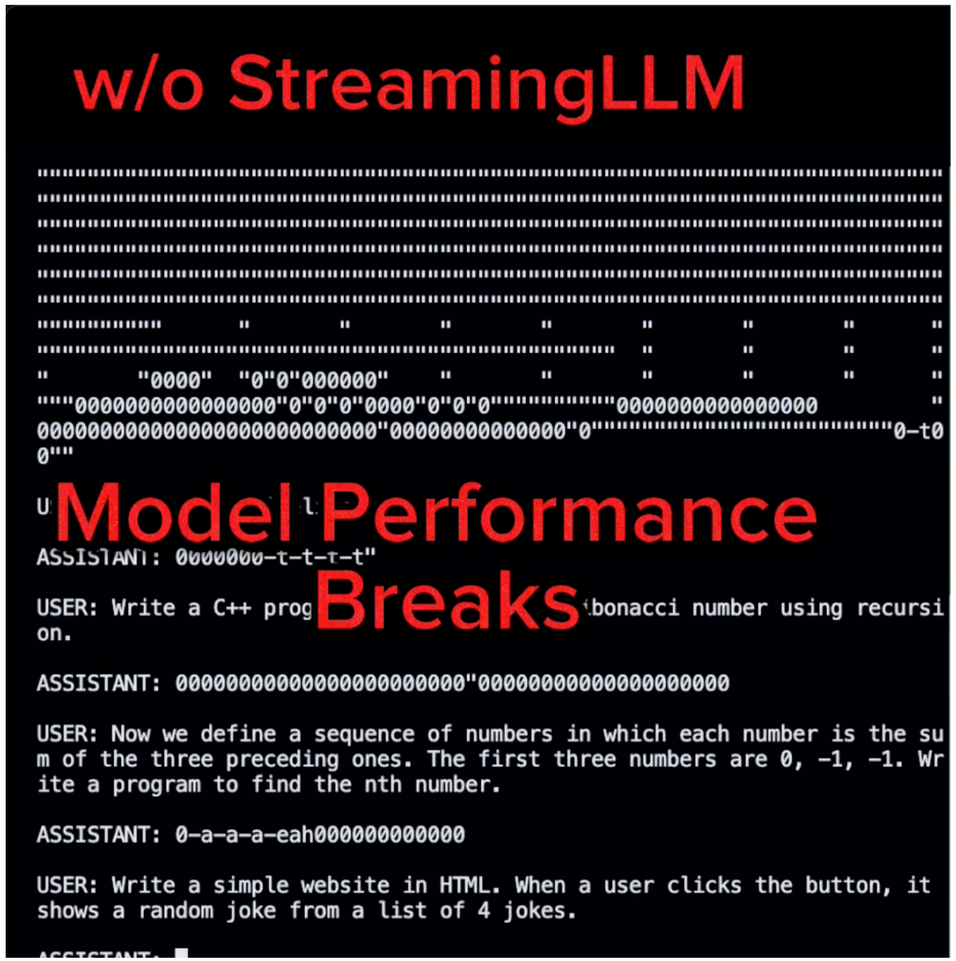


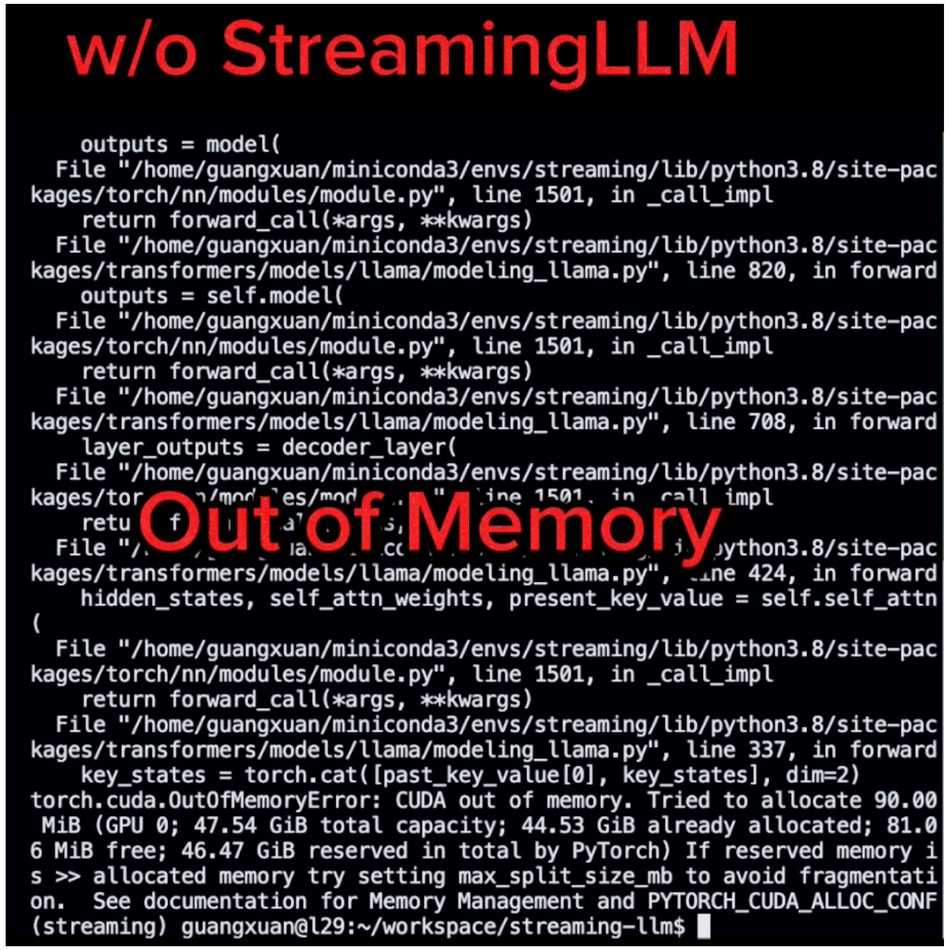
https://github.com/tomaarsen/attention_sinks

Challenges of Deploying LLMs in Streaming Applications



Challenges of Deploying LLMs in Streaming Applications

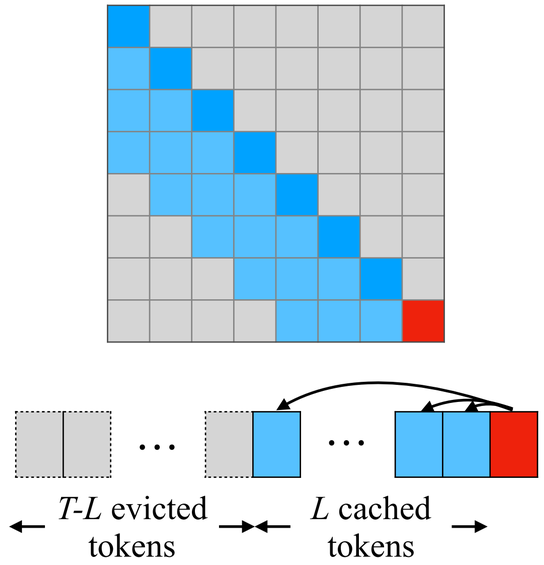




The Limits of Window Attention

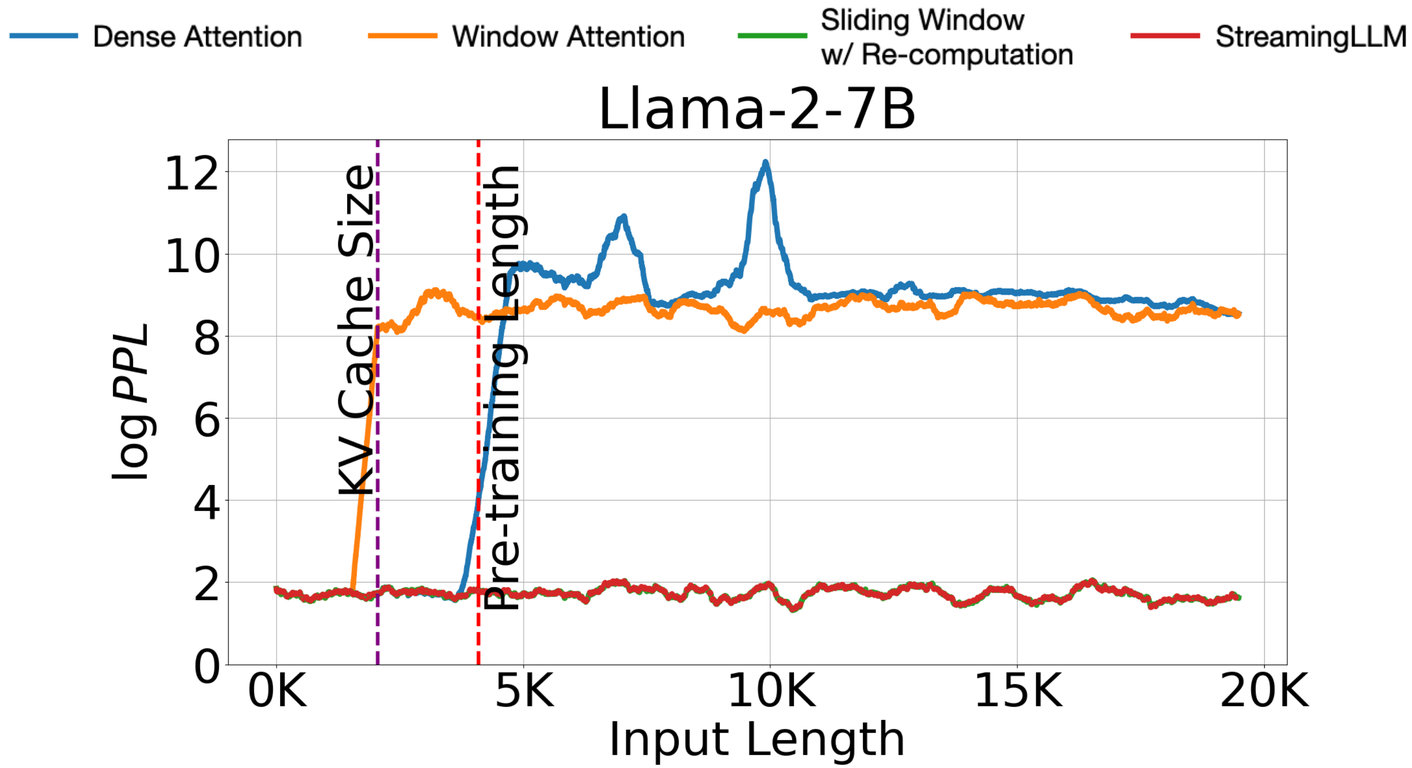
- A natural approach — window attention: caching only the most recent Key-Value states.
- Drawback: model collapses when the text length surpasses the cache size, when the initial token is evicted.

(b) Window Attention



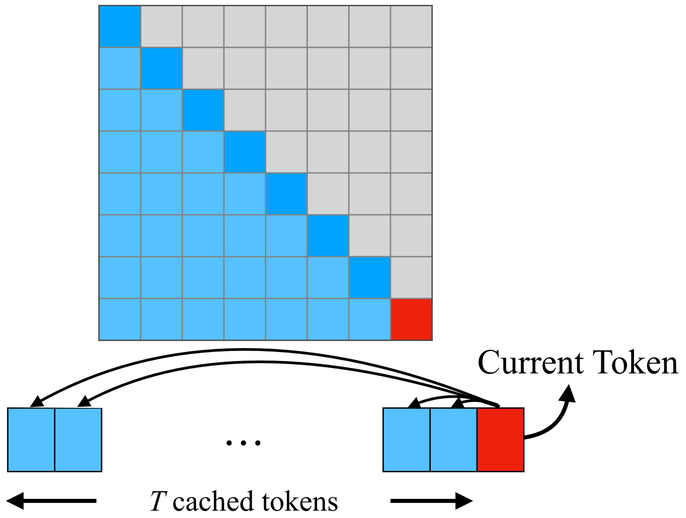
$O(TL)$ ✔ **PPL:** 5158

Breaks when initial tokens are evicted.



Difficulties of Other Methods

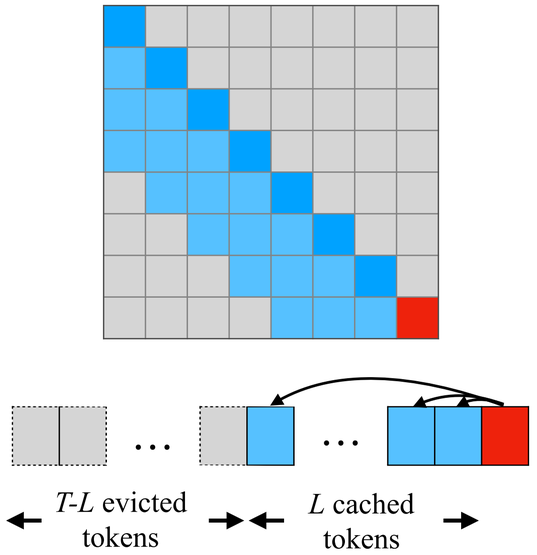
(a) Dense Attention



$O(T^2)$ ✗ **PPL:** 5641 ✗

Has poor efficiency and performance on long text.

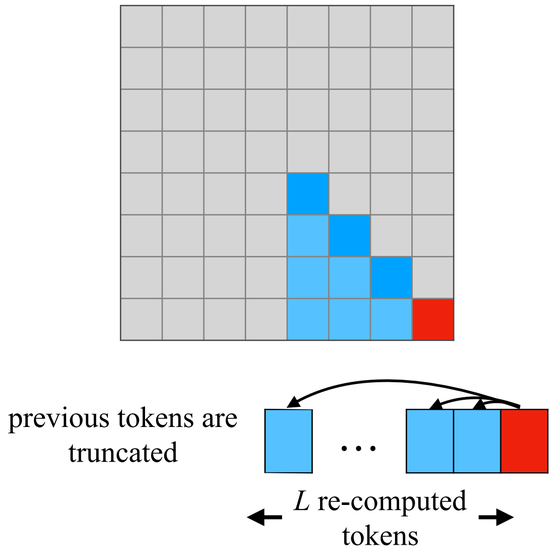
(b) Window Attention



$O(TL)$ ✓ **PPL:** 5158 ✗

Breaks when initial tokens are evicted.

(c) Sliding Window w/ Re-computation



$O(TL^2)$ ✗ **PPL:** 5.43 ✓

Has to re-compute cache for each incoming token.

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.

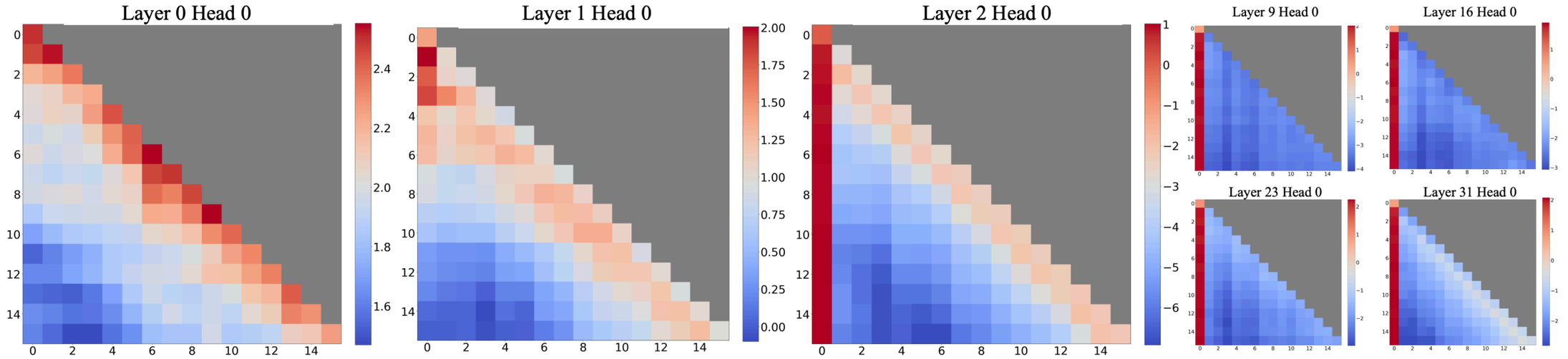


Figure 2: Visualization of the *average* attention logits in Llama-2-7B over 256 sentences, each with a length of 16. Observations include: (1) The attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern, with recent tokens receiving more attention. (2) Beyond the bottom two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

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

The “Attention Sink” Phenomenon

- We observed the phenomenon in the SpAtten project in 2021, but wan’t able to explain it until 2023:

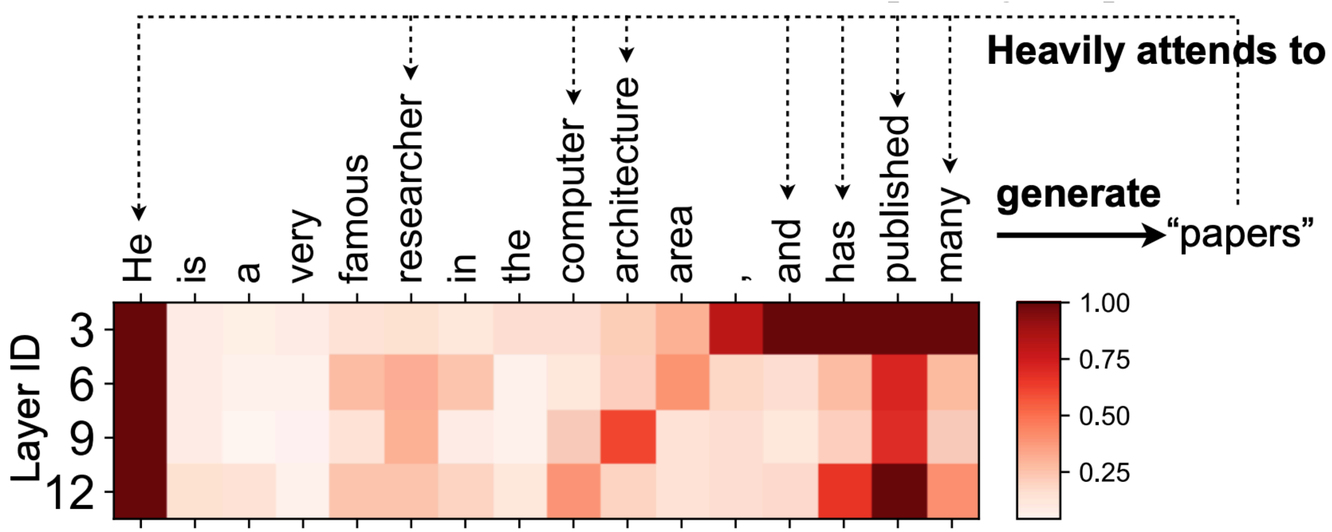


Fig. 23. Cumulative importance scores in GPT-2. Unimportant tokens are pruned on the fly. Important tokens are heavily attended.

GPT-2 for Language Modeling

```
Du Fu was a great poet of the Tang dynasty. Recently a variety of styles have been used in efforts to translate the work of Du Fu into English
Du ~~Fu was a great~~ poet ~~of the~~ Tang dynasty. Recently ~~a variety of~~ styles ~~have been used~~ in efforts to translate the work of Du Fu into English
Du ~~Fu was a great poet of the Tang dynasty. Recently a variety of styles have been used in efforts to~~ translate ~~the work of Du Fu~~ into English
```

‘*English*’ is the generated token.

SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning, HPCA’21

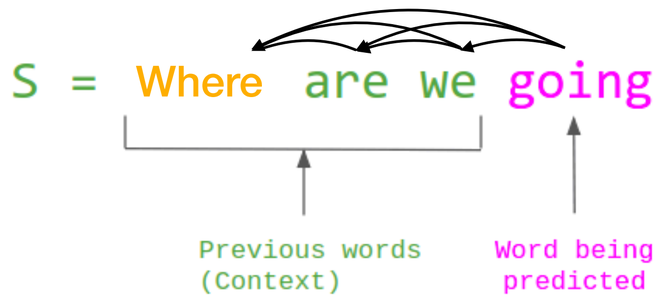
Understanding Why Attention Sinks Exist

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.

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

- 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)

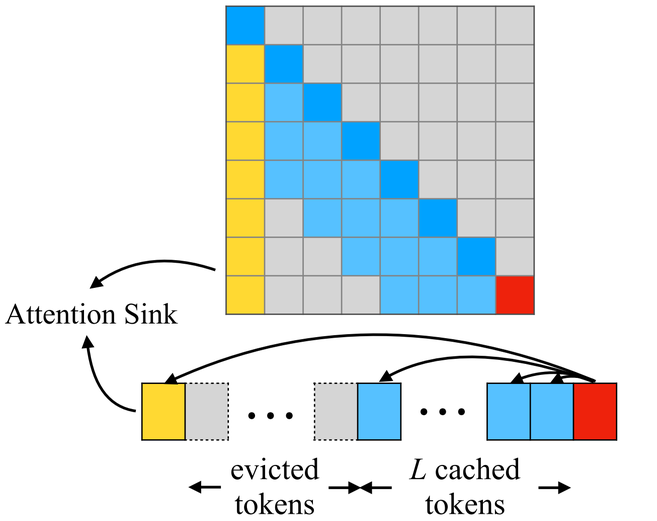
- Does the importance of the initial tokens arise from their **position** or their **semantics**?
  - We found adding initial four “\n”s can also recover perplexity.
  - Therefore, it is **position**!

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

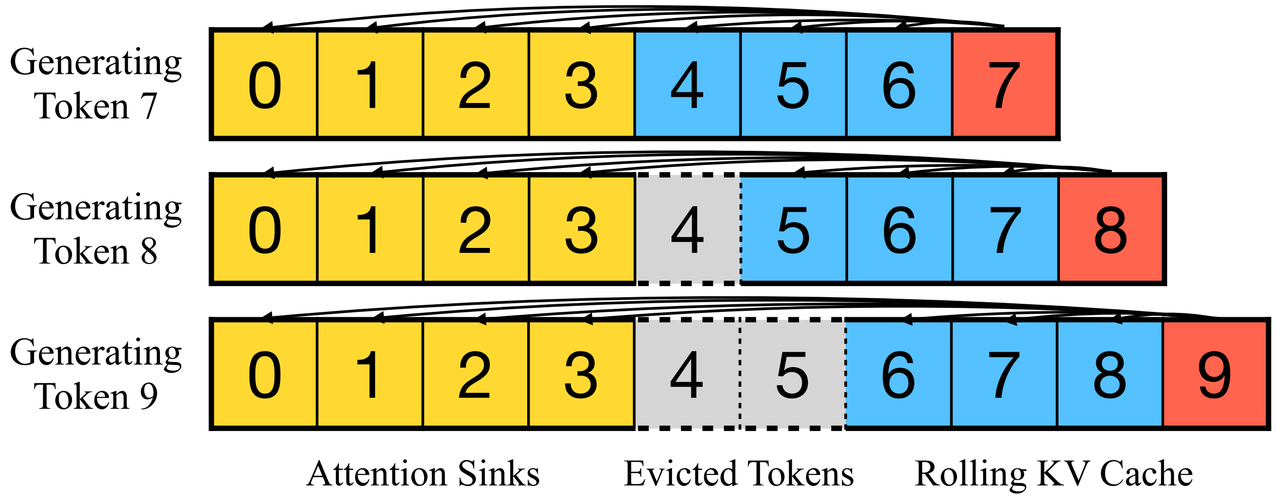
- **Objective:** Enable LLMs trained with a finite attention window to handle infinite text lengths without additional training.
- **Key Idea:** **preserve the KV of attention sink tokens**, along with the sliding window's KV to stabilize the model's behavior.

(d) **StreamingLLM (ours)**



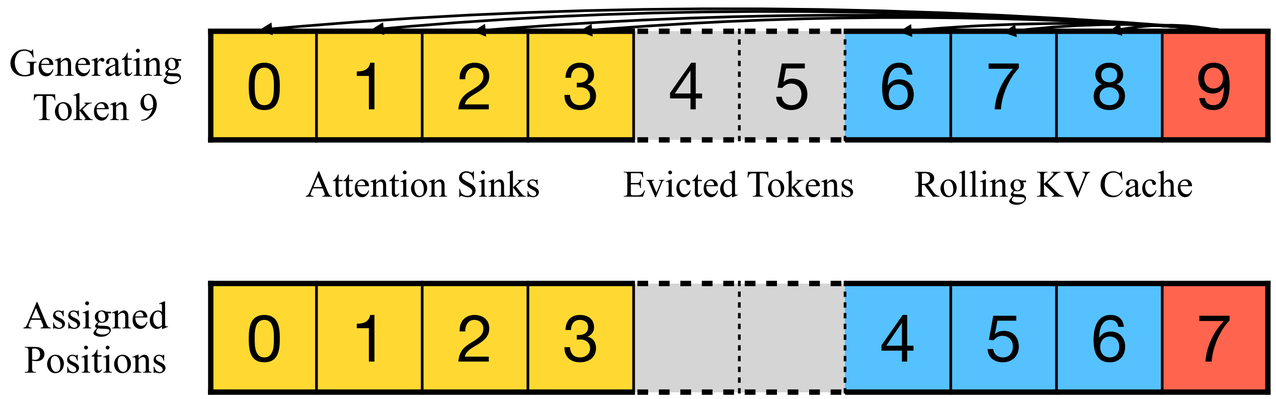
$O(TL)$ ✔ **PPL:** 5.40 ✔

Can perform efficient and stable language modeling on long texts.



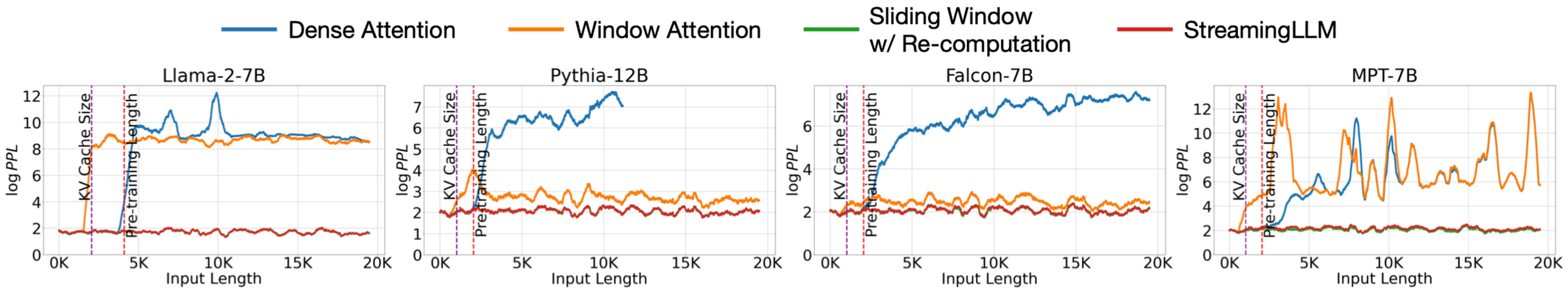
Positional Encoding Assignment

- 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.

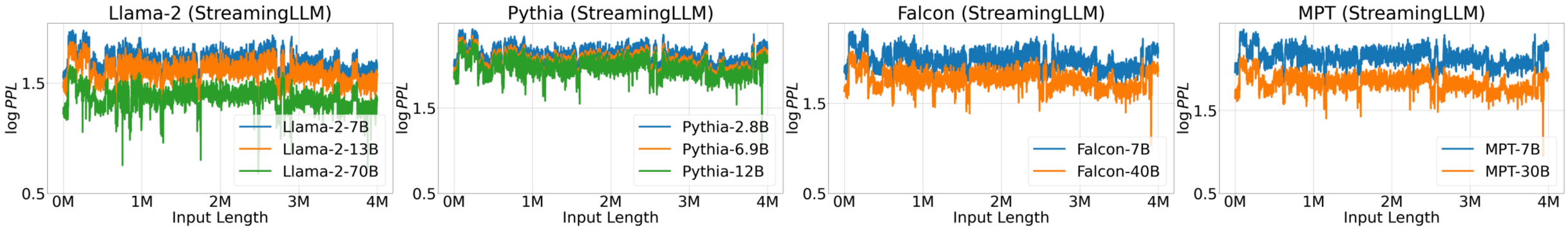


- **Dense attention** fails beyond pre-training attention window size.
- **Window attention** fails after input exceeds cache size (initial tokens evicted).
- **StreamingLLM** shows stable performance; perplexity close to sliding window with re-computation baseline.

Streaming Performance

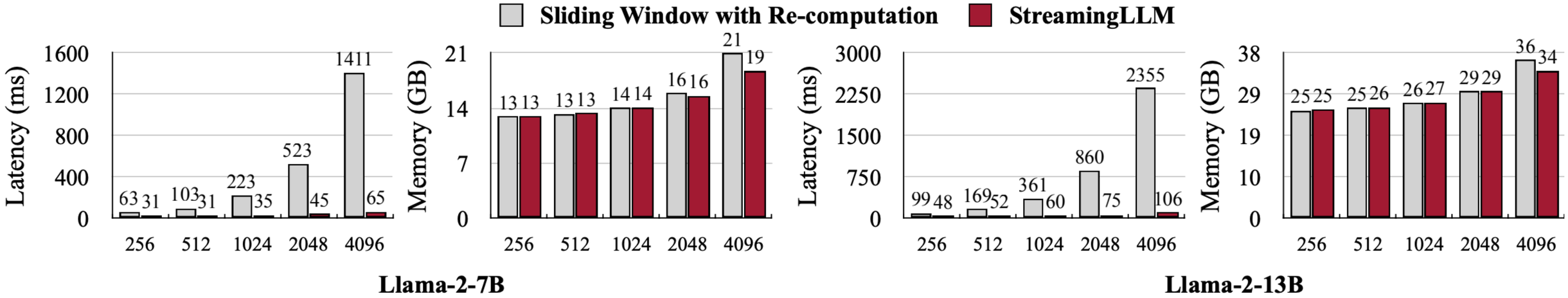
Super Long Language Modeling

- 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.

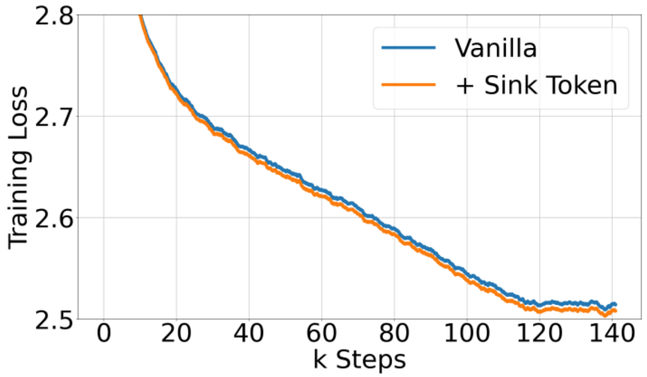
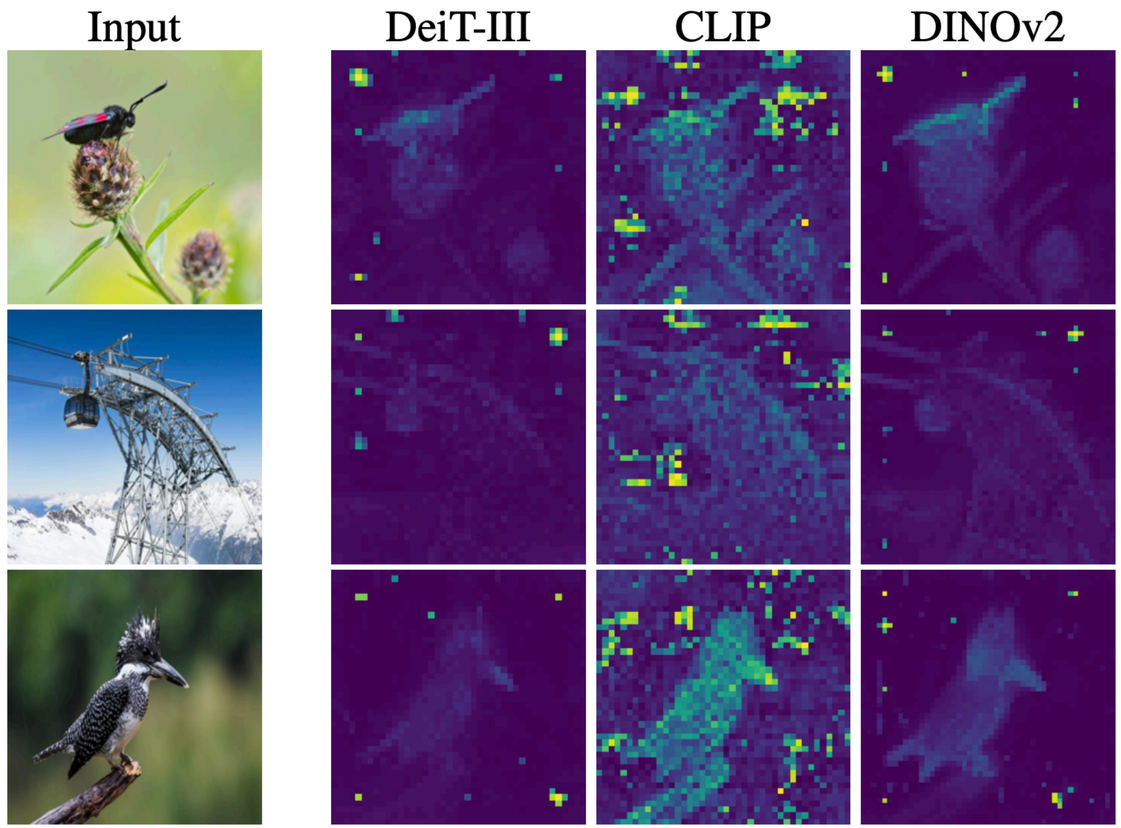


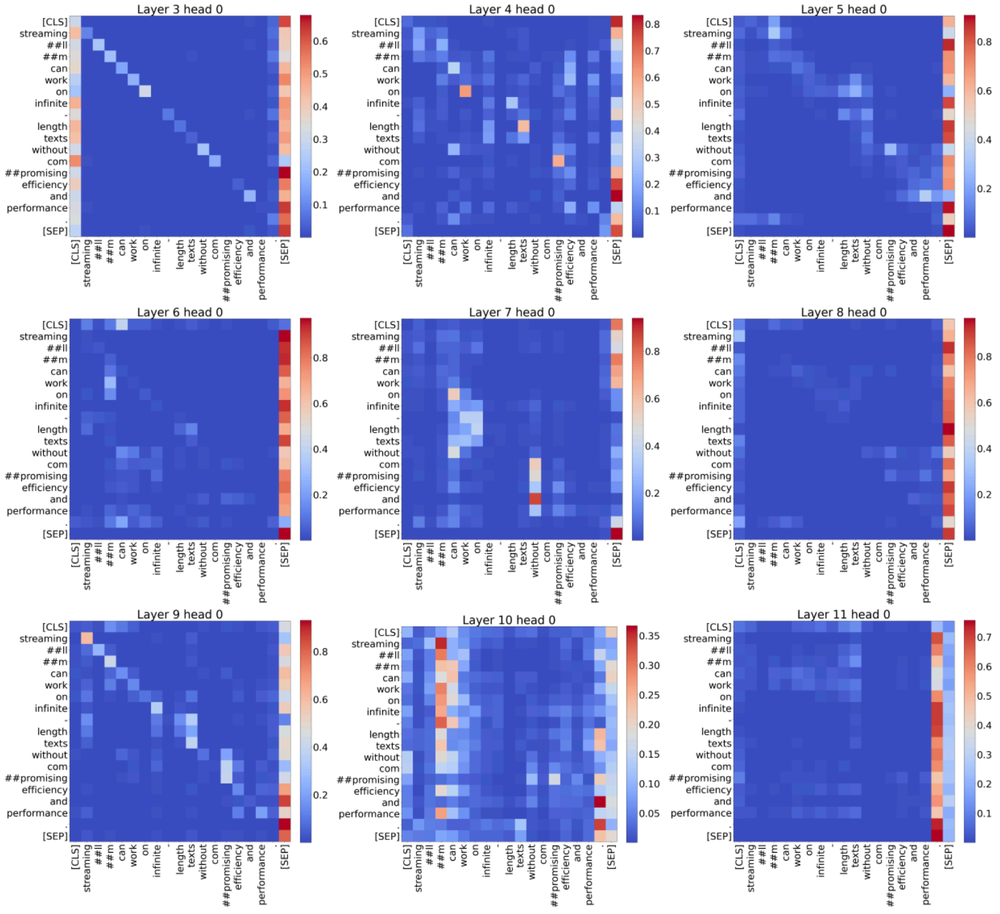
Figure 6: Pre-training loss curves of models w/ and w/o sink tokens. Two models have a similar convergence trend.

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: attention sinks emerge in low semantic background pixels.

BERT: attention sinks are [SEP] tokens at the end of all sentences

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

exceed

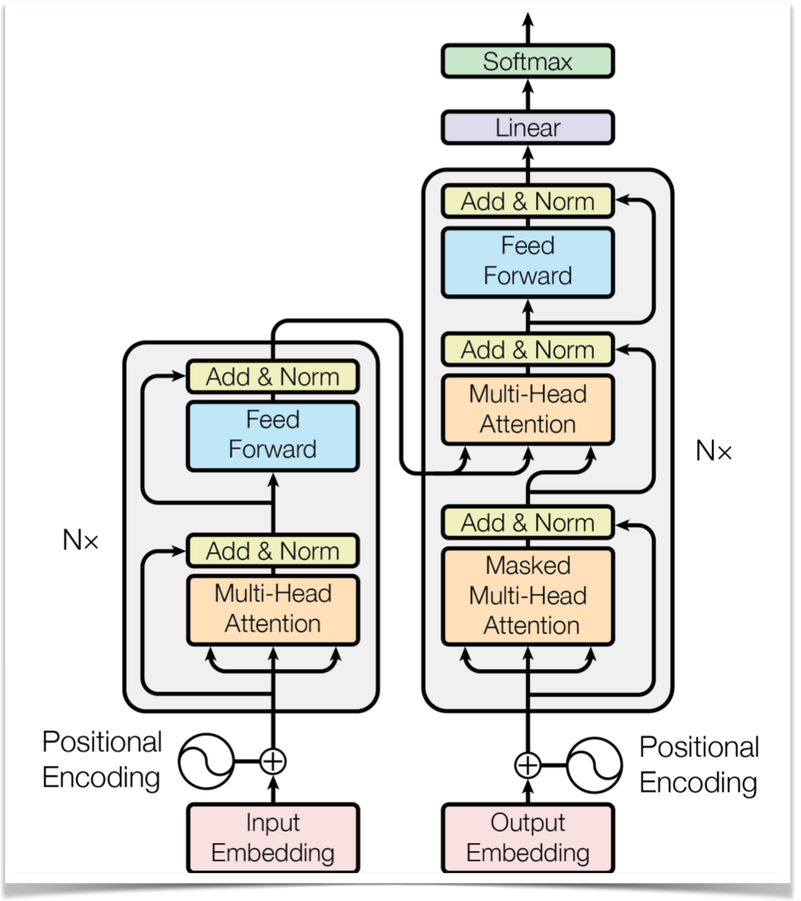
drop to zero

How to solve this issue?

Lecture Plan

Today, we will cover:

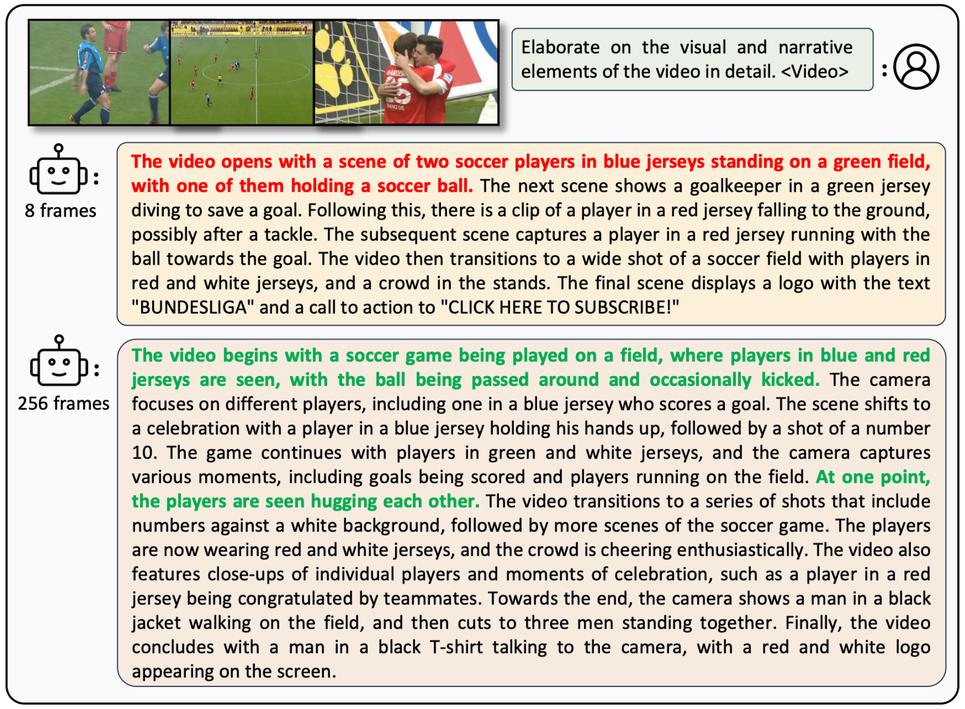
1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



DuoAttention

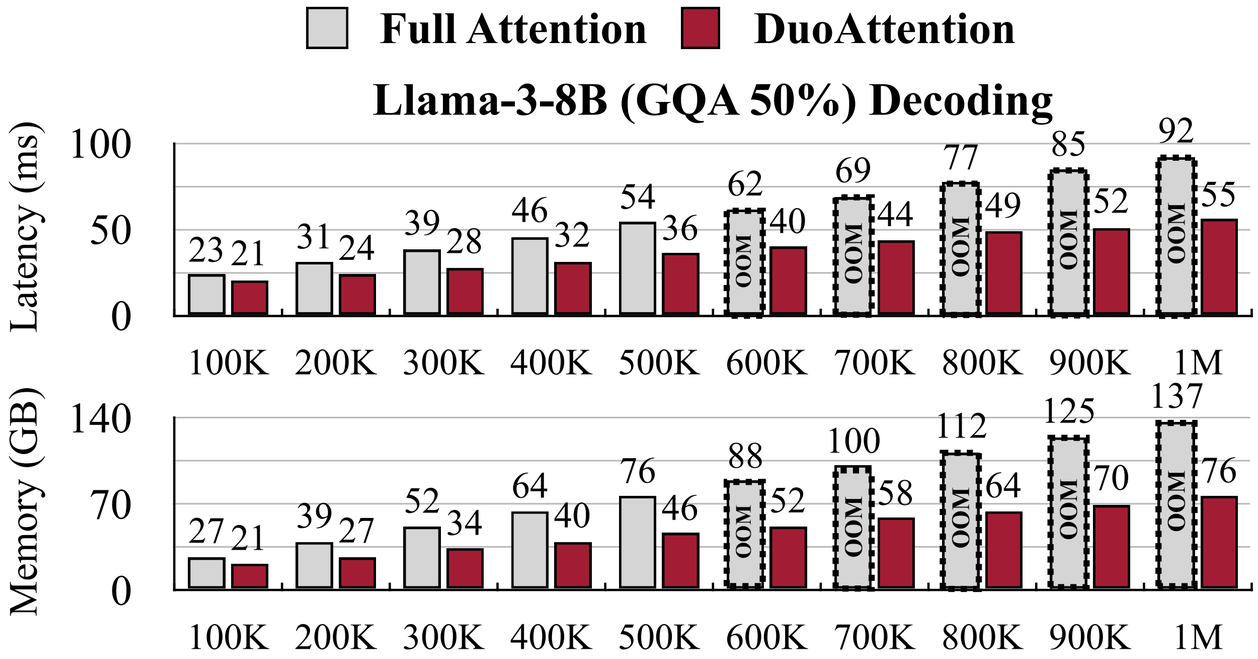
Deploying Long-Context LLMs is Crucial But Challenging

- LLMs need to handle long-context like summarizing long texts and processing images/videos.
- Memory and latency increase dramatically with context length.



a 224×224 image = 256 tokens

a 1-hour video at 1 FPS = **1 million tokens**

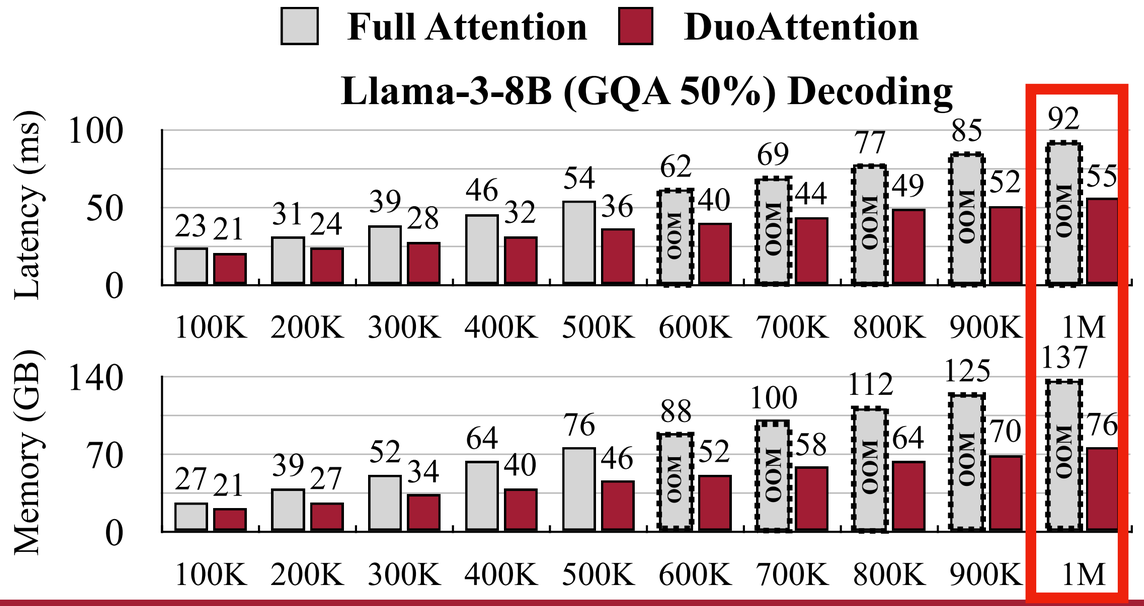


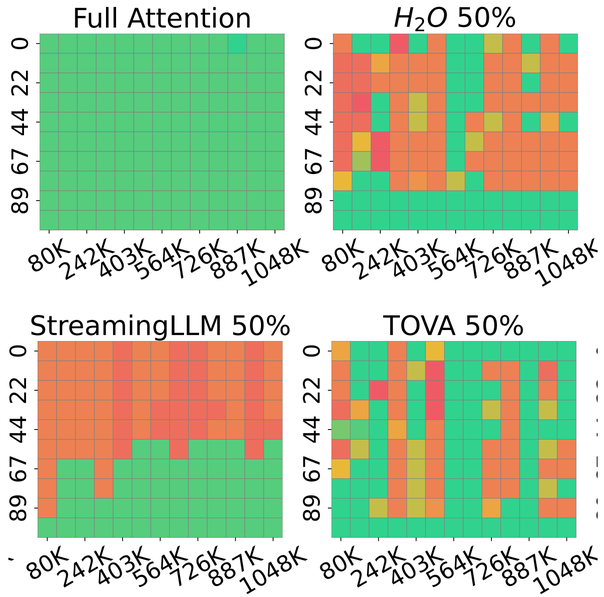
LongVILA: Scaling Long-Context Visual Language Models for Long Videos

Key Challenges

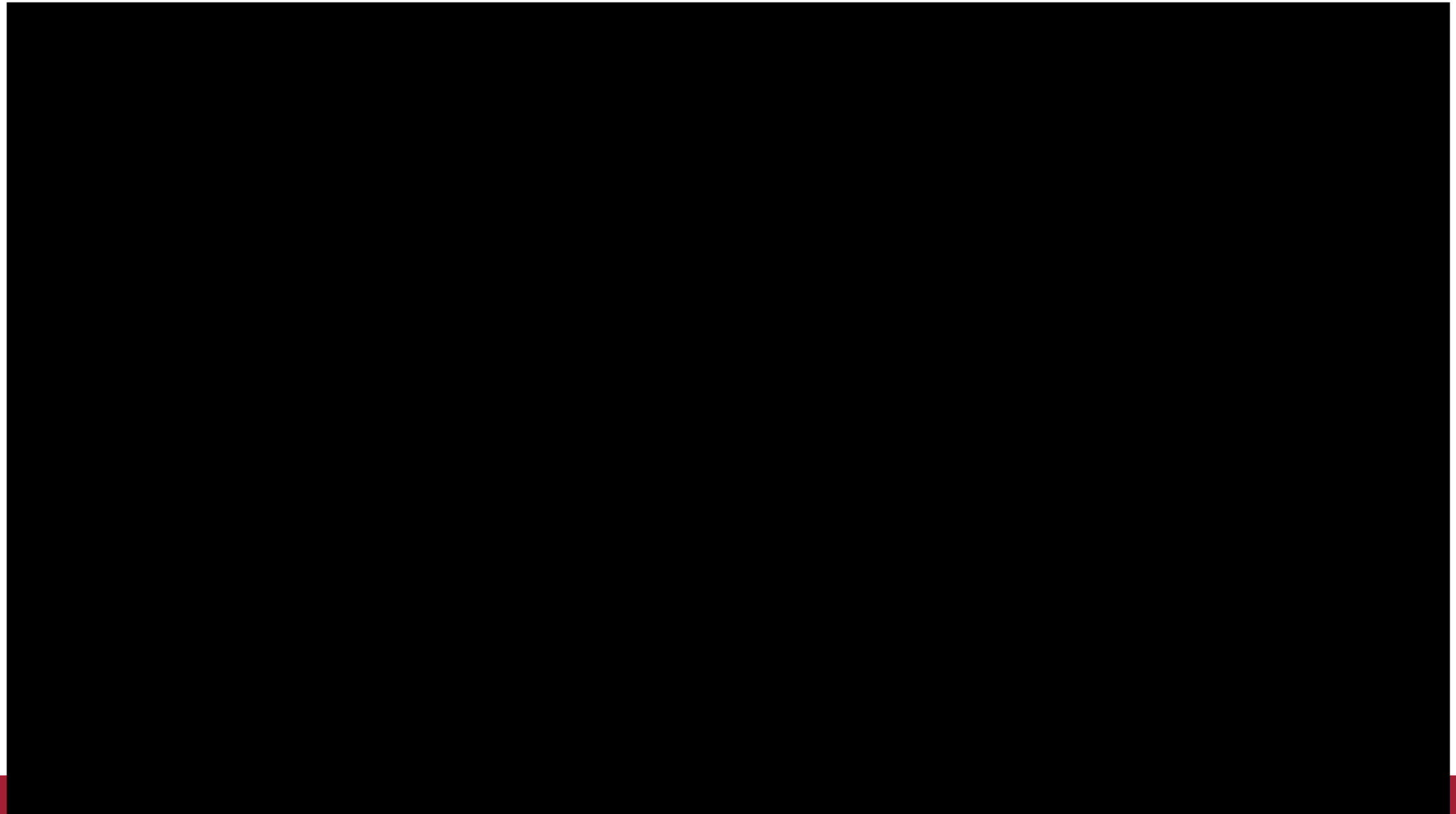
Memory, Latency, and Long-Context Accuracy

- **Memory Bottleneck:** Storing KV states across all tokens consumes massive memory (e.g., 137 GB for 1 million context in Llama-3-8B).
- **Decoding Latency:** Decoding time grows linearly with sequence length (e.g., 92 ms per token for 1 million context in Llama-3-7B).
- Existing KV cache compression methods **damages** LLMs' long-context ability.





Running LLMs with 3.3 Million Contextual Tokens on an A100 GPU

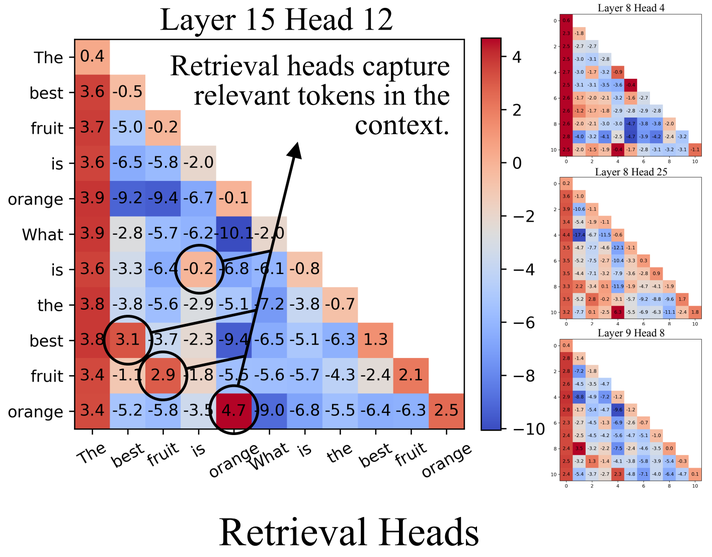


Retrieval vs. Streaming Heads

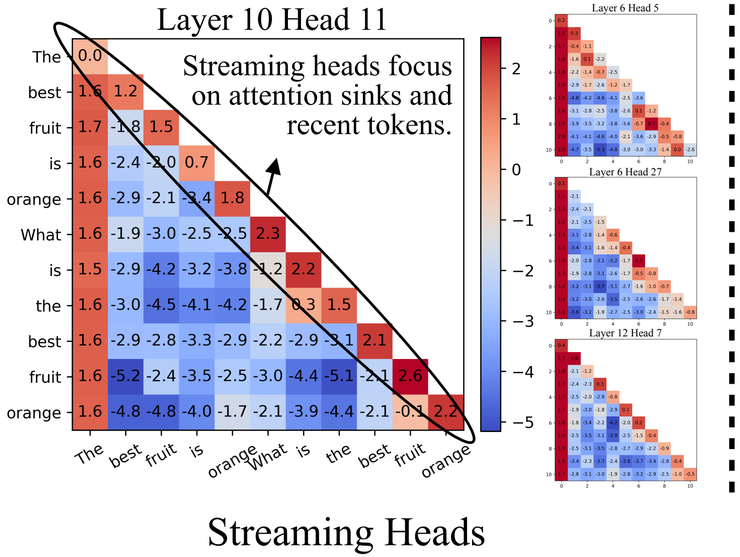
Retrieval Heads vs. Streaming Heads

- **Retrieval Heads:**
  - Capture contextually important tokens from earlier in the sequence.
  - Require full attention across all tokens in the context.
  - Compressing their KV cache would cause a significant loss in performance.

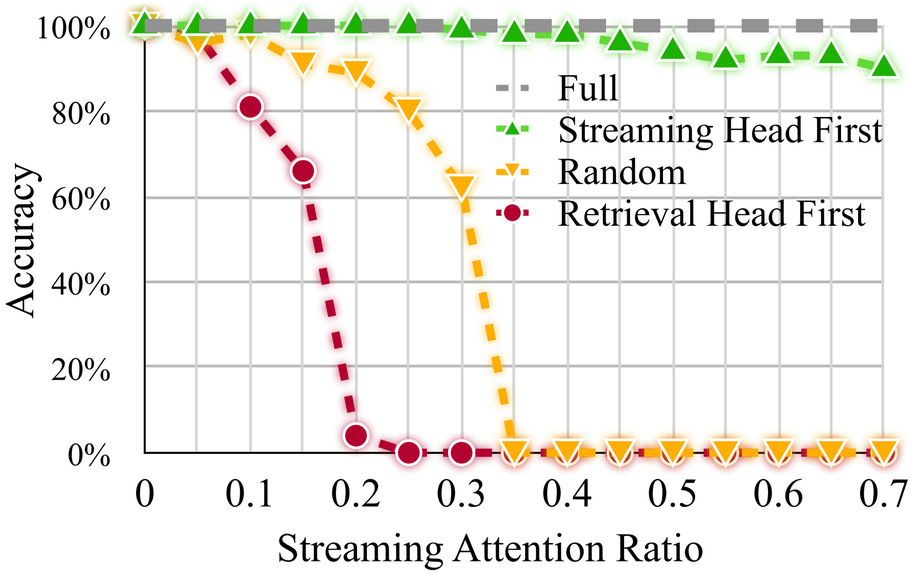
- **Streaming Heads:**
  - Focus on recent tokens and attention sinks.
  - Use a reduced KV cache that only stores recent tokens.
  - Reducing their cache size has minimal impact on performance.



Retrieval Heads



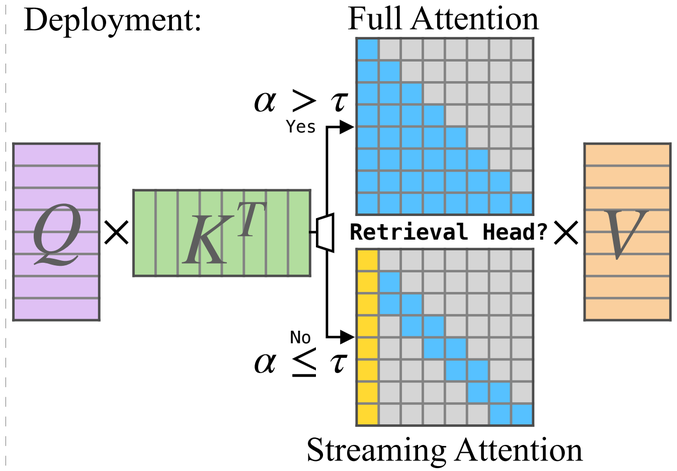
Streaming Heads

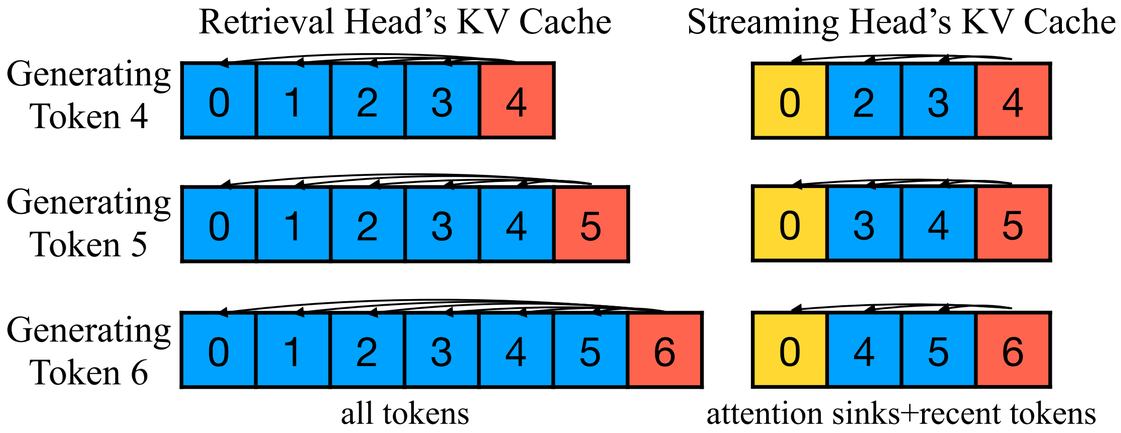


Streaming Attention Ratio

DuoAttention Framework Overview

- **Key Insight: Not all attention heads in a model need full attention across the entire context.**
  - **Retrieval Heads** focus on important tokens from earlier in the sequence and need full KV cache.
  - **Streaming Heads** focus on recent tokens and attention sinks, needing only a reduced KV cache.
- **Solution**: DuoAttention only applies full KV cache to Retrieval Heads, and uses lightweight, constant-length caches for Streaming Heads. Reduces memory and decoding latency while preserving long-context capabilities.



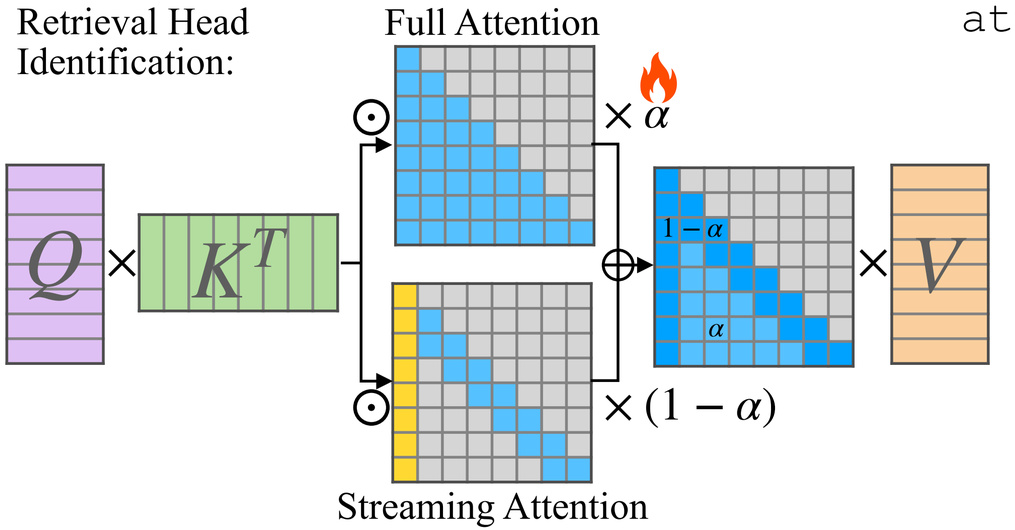


DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads [Xiao et al., 2024]

Optimization-Based Identification of Retrieval Heads

Efficiently and Accurately identify Retrieval Heads

- **Challenge:** Precisely identifying which heads are critical for long-context processing is difficult.
- **Optimization-Based Approach:**
  - Assign a trainable gate value (α) to each head.
  - The gate value blends full attention and streaming attention, allowing the model to learn which heads are necessary.
  - Minimize output deviation from the full attention model to maintain accuracy.



$$\texttt{attn}_{i,j} = \alpha_{i,j} \cdot \texttt{full\_attn} + (1 - \alpha_{i,j}) \cdot \texttt{streaming\_attn}$$

$$\texttt{full\_attn} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T \odot \boldsymbol{M}_{\text{causal}})\boldsymbol{V},$$

$$\texttt{streaming\_attn} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T \odot \boldsymbol{M}_{\text{streaming}})\boldsymbol{V},$$

$$\mathcal{L}_{\text{distill}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=T-l+1}^{T}(\boldsymbol{H}_{\text{full}}^{(i)}[j] - \boldsymbol{H}_{\text{mixed}}^{(i)}[j])^2$$

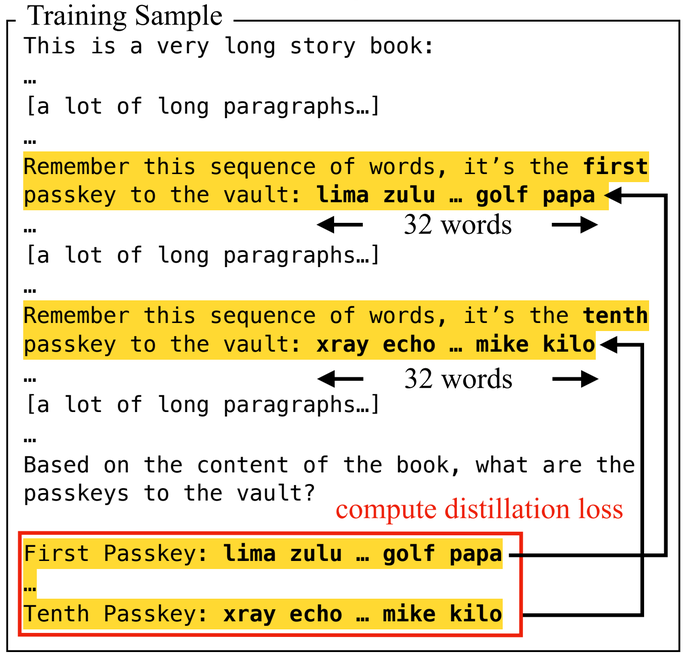
$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^{L}\sum_{j=1}^{H}|\alpha_{i,j}|$$

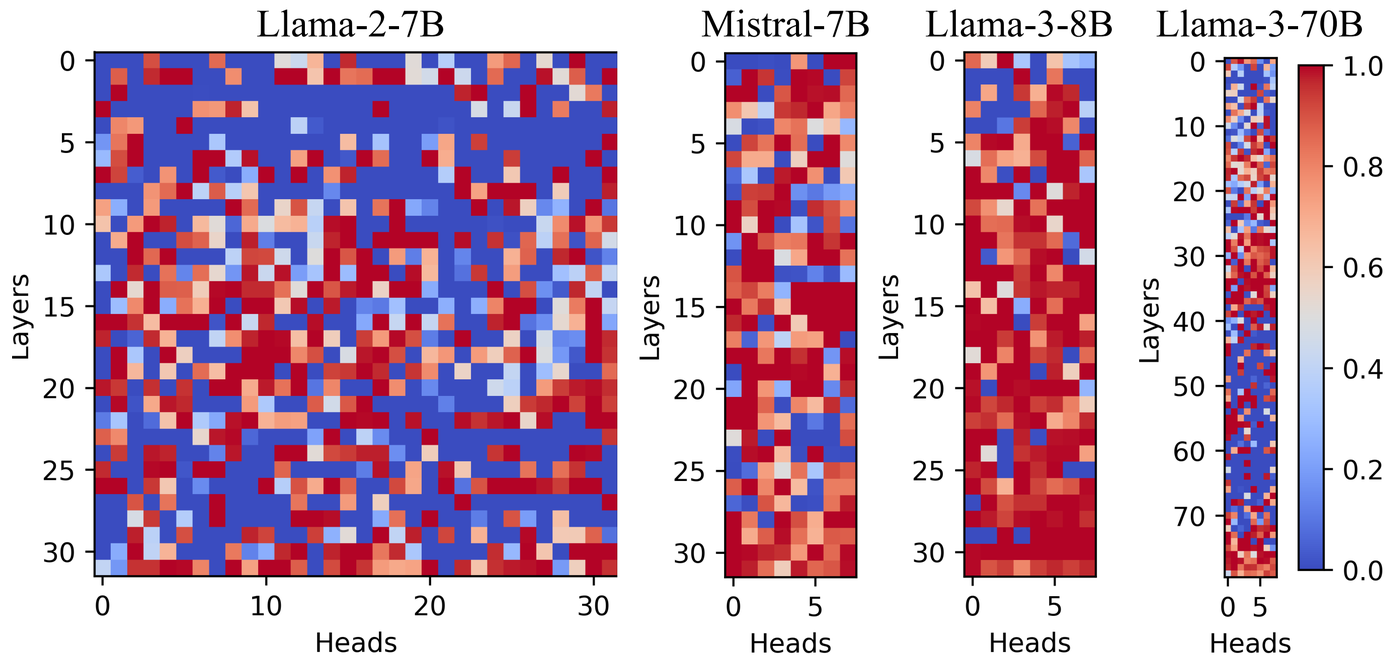
$$\mathcal{L} = \mathcal{L}_{\text{distill}} + \lambda\mathcal{L}_{\text{reg}}$$

Optimization-Based Identification of Retrieval Heads

Using Synthetic Data to Focus on LLMs' Retrieval Ability

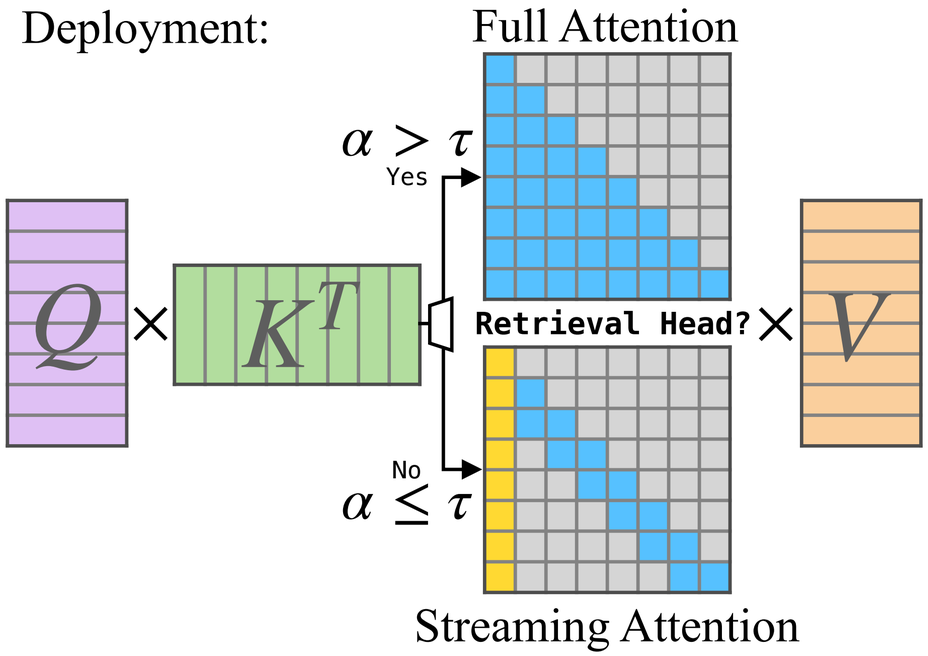
- **Synthetic Dataset:**
  - Embeds ten passkeys in a long context.
  - LLM recalls the passkeys to identify which heads need full attention for long-context retrieval.
- **Efficiency:**
  - Only ~1K gate values need to be trained. (e.g. 32 layers x 32 heads for llama2-7B)
  - The process is completed in a few hours on 8 A100 GPUs.

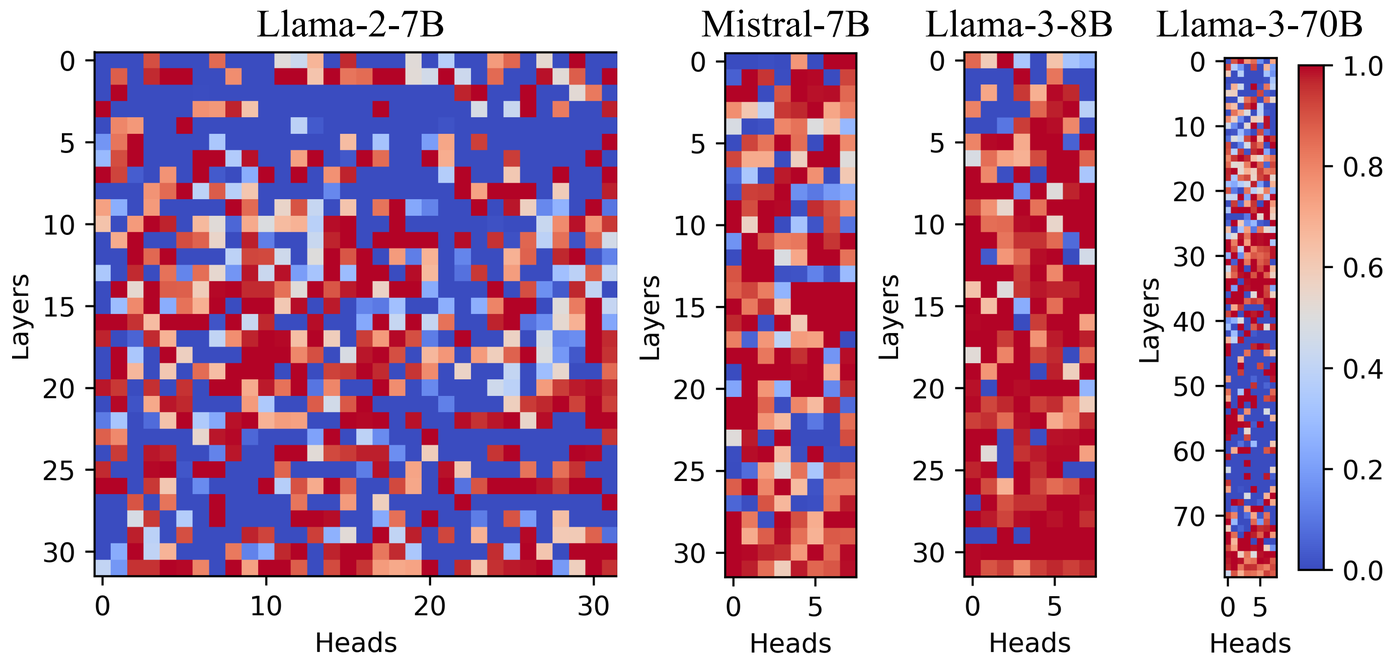




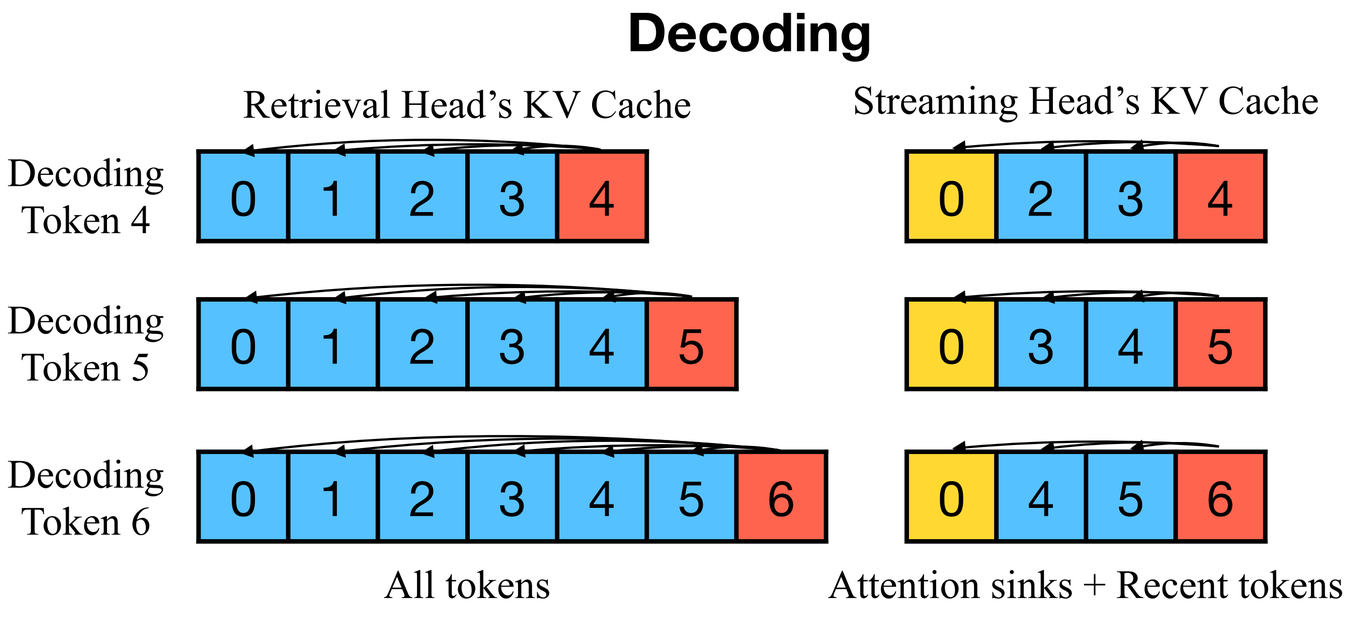
Deploying LLMs with DuoAttention

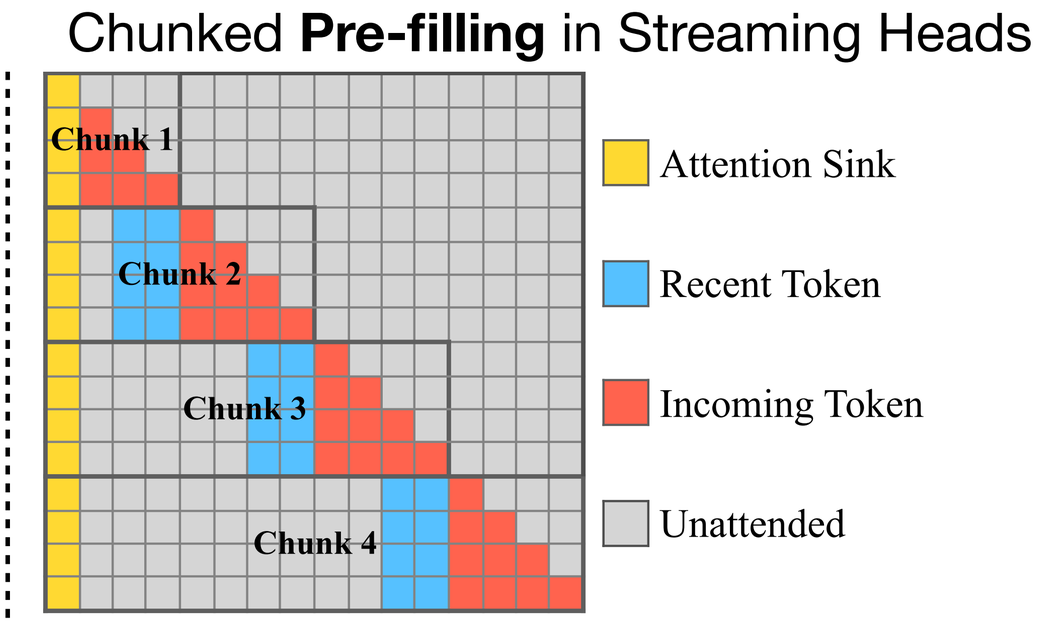
- **Binarizing Attention: During deployment, the trained gate values (α) are binarized to classify each head as either a Retrieval Head or a Streaming Head.**
- **Reordering Attention Heads:**
  - During deployment, attention heads are reordered into distinct clusters for efficient processing.
  - This allows for better slicing and concatenation of KV caches, improving inference speed.





Decoding and Chunked Pre-filling of DuoAttention





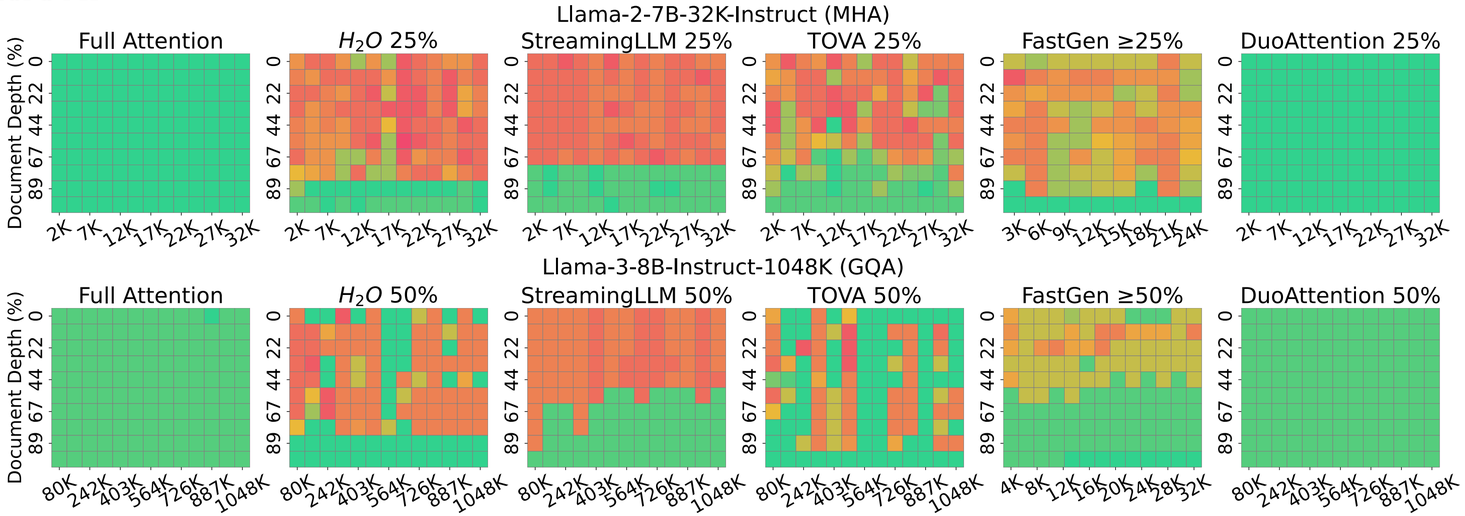
- **Two KV Caches:**
  - **Retrieval Head:** Stores all tokens.
  - **Streaming Head:** Stores attention sinks and recent tokens (constant memory).
- **Computation:**
  - Queries, keys, and values are split into retrieval and streaming heads.
  - Outputs from both heads are concatenated for the final projection.

- **Chunked Pre-filling splits sequence into fixed-length chunks.**
- **Streaming head's KV cache keeps only sink and recent tokens after each pre-filling chunk.**
- **Pre-filling Complexity on Streaming Heads:**
  - Time complexity reduced from $O(L^2)$ to $O(LK)$.
  - Memory complexity reduced from $O(L)$ to $O(K)$
  - L is sequence length and K is chunk size.

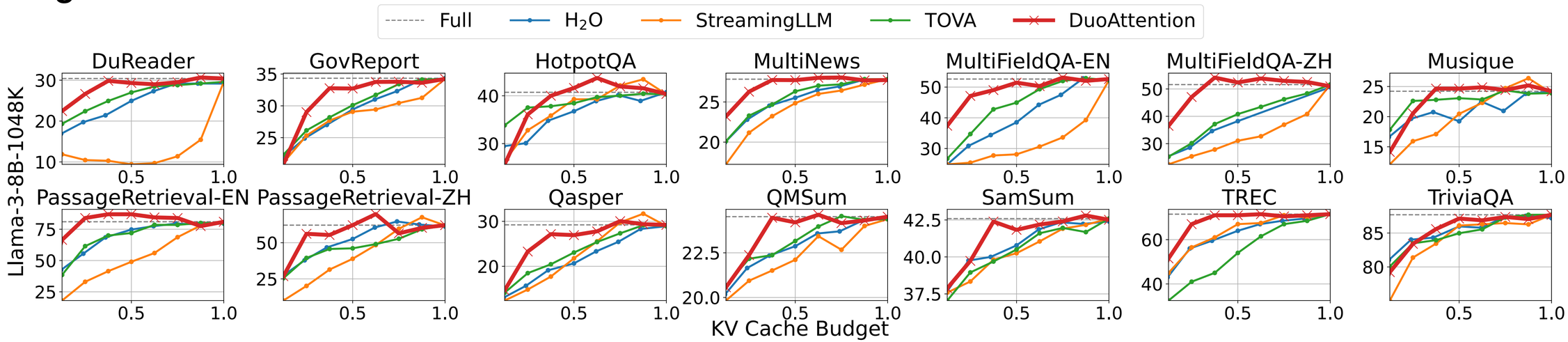
Results on Accuracy Benchmarks

Long-Context Benchmarks: Needle-in-a-Haystack and LongBench

- **Needle-in-a-Haystack:**



- **LongBench:**



Results on Accuracy Benchmarks

Short-Context Benchmarks: MBPP, MMLU, and MT-Bench

- **DuoAttention doesn't harm LLMs other general abilities.**

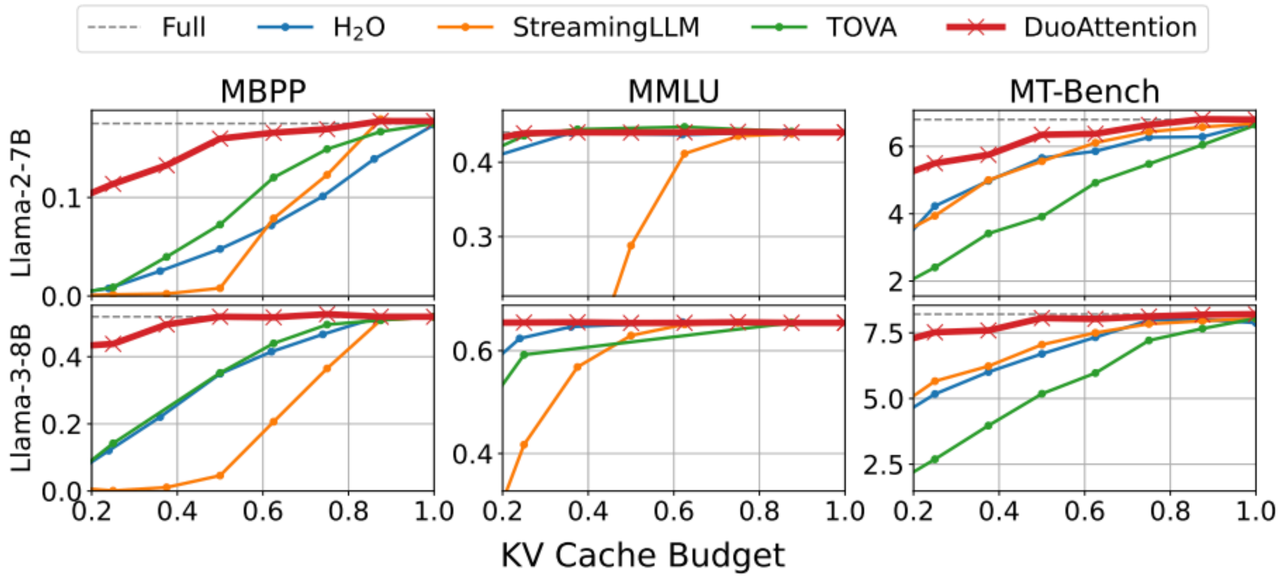


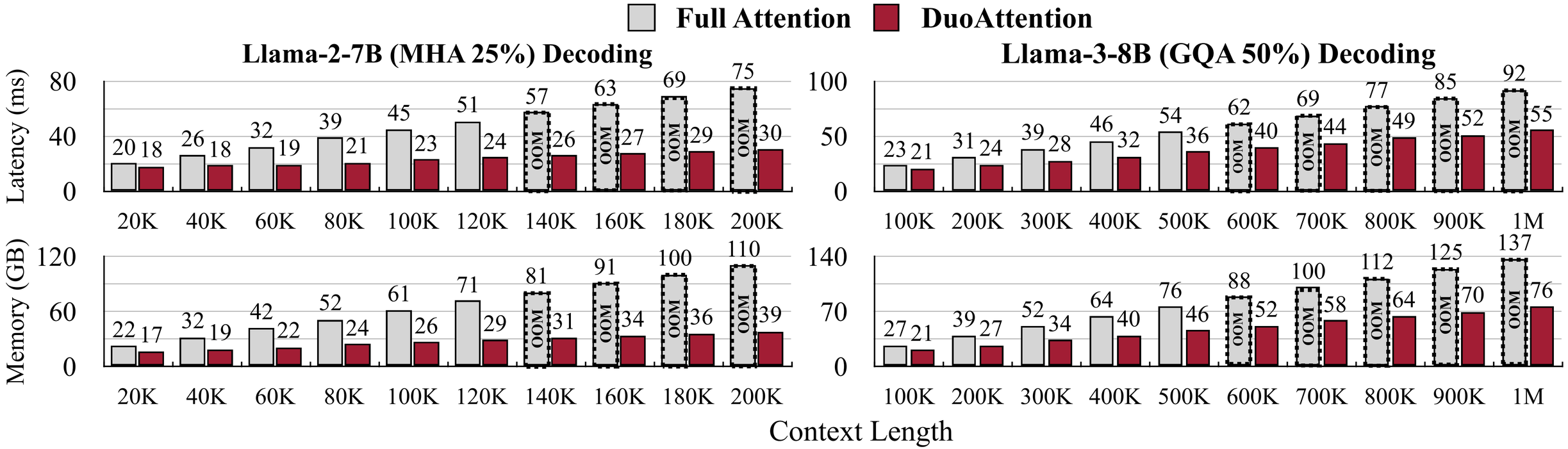
Figure 8: Results on short benchmarks.

Table 1: Llama-3-70B results on short benchmarks.

	Budget	MMLU	MBPP	MT-B
Full	100%	79.38%	47.85%	8.93
H2O	50%	79.26%	32.12%	7.16
TOVA	50%	79.15%	36.09%	7.96
SLLM	50%	77.46%	5.57%	5.41
DuoAttn	50%	**79.35%**	**47.09%**	**9.14**

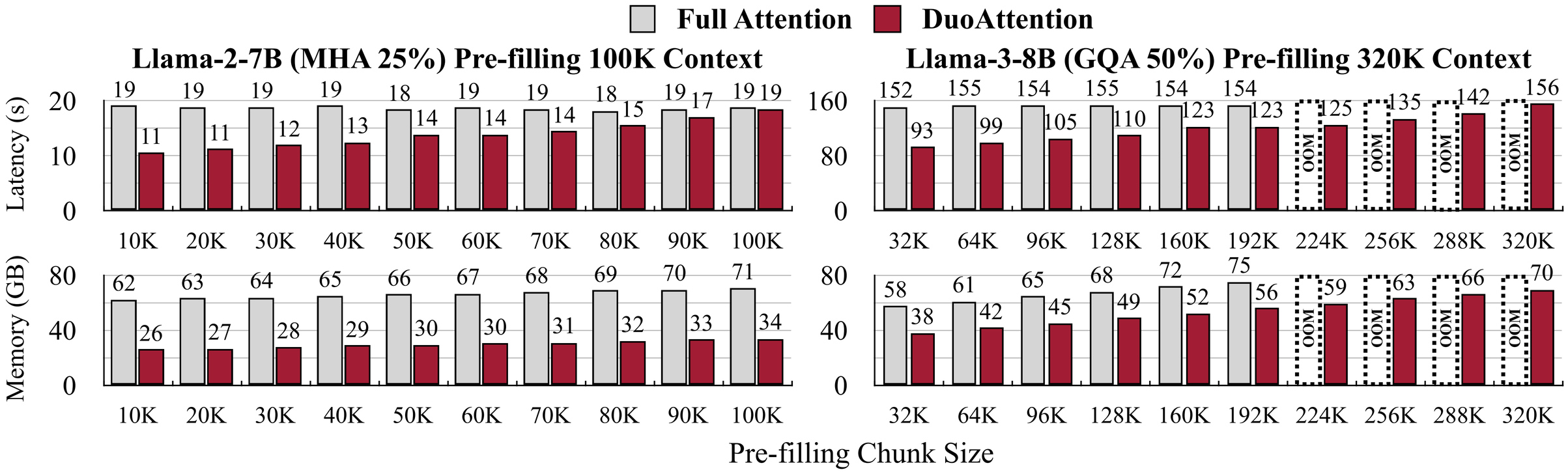
Decoding Memory and Latency Improvements

- **DuoAttention provides up to 2.45× memory reduction for MHA and 1.65× for GQA models, and up to 2.13× decoding latency improvement for MHA and 1.5× for GQA models.**



Pre-filling Memory and Latency Improvements

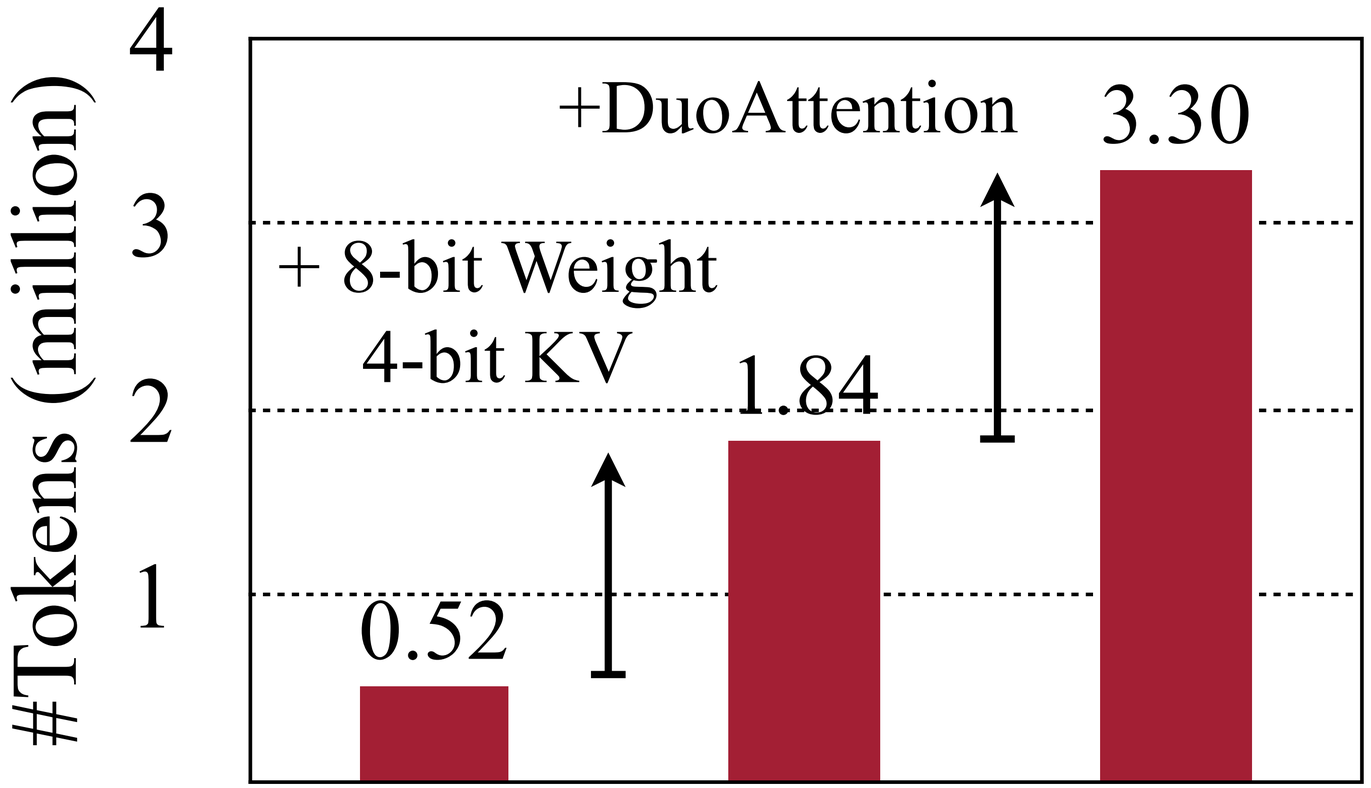
- **DuoAttention achieves up to 1.73x pre-filling latency reduction for MHA and 1.63x for GQA models, with memory reductions up to 2.38x for MHA and 1.53x for GQA models.**



Combination with KV Cache Quantization

Pushing the Limits of Context Length on Memory-constrained Devices

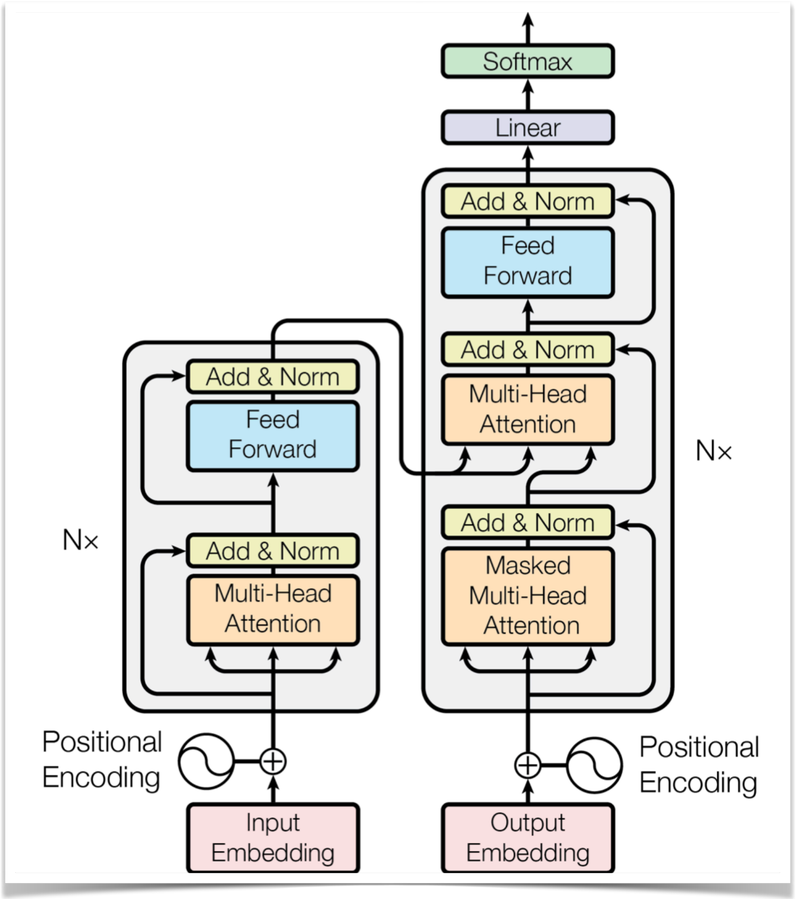
- **Up to 3.3 million tokens can be handled in a single A100 GPU using DuoAttention combined with 4-bit KV cache quantization.**



Lecture Plan

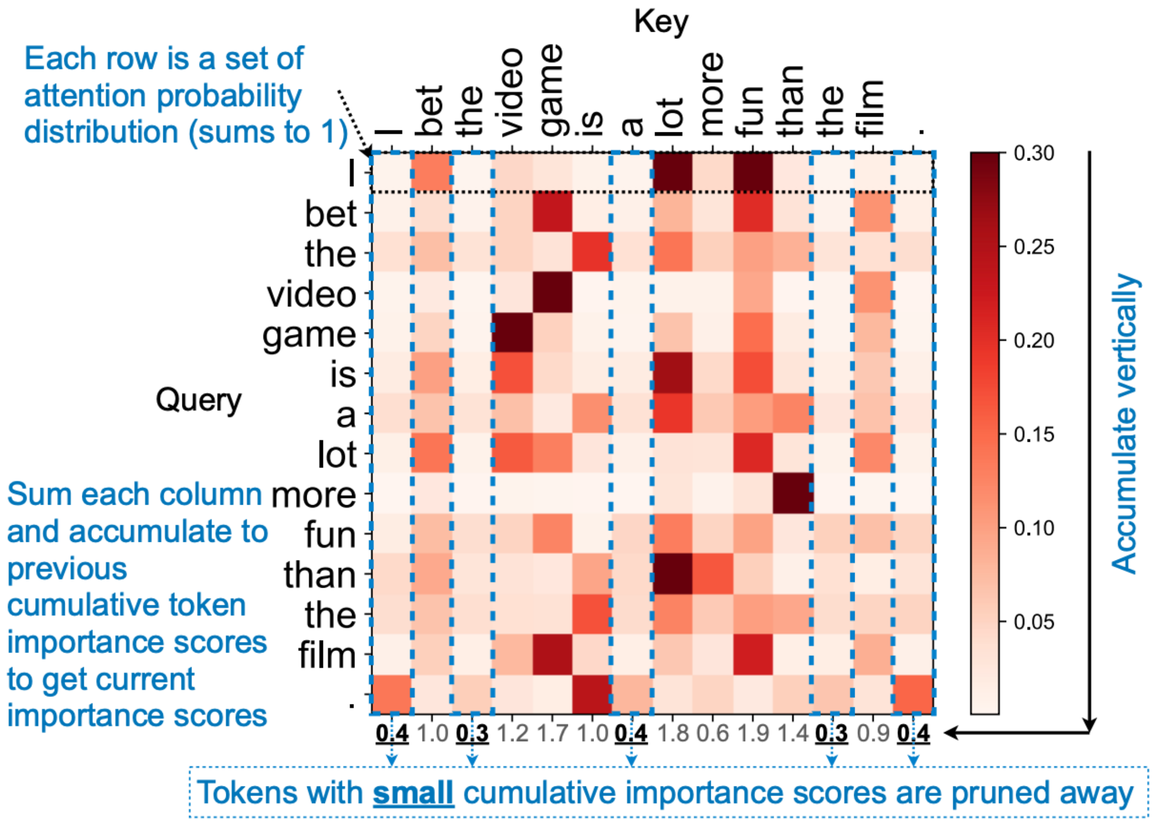
Today, we will cover:

1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. Beyond Transformers
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



SpAtten: Sparse Attention

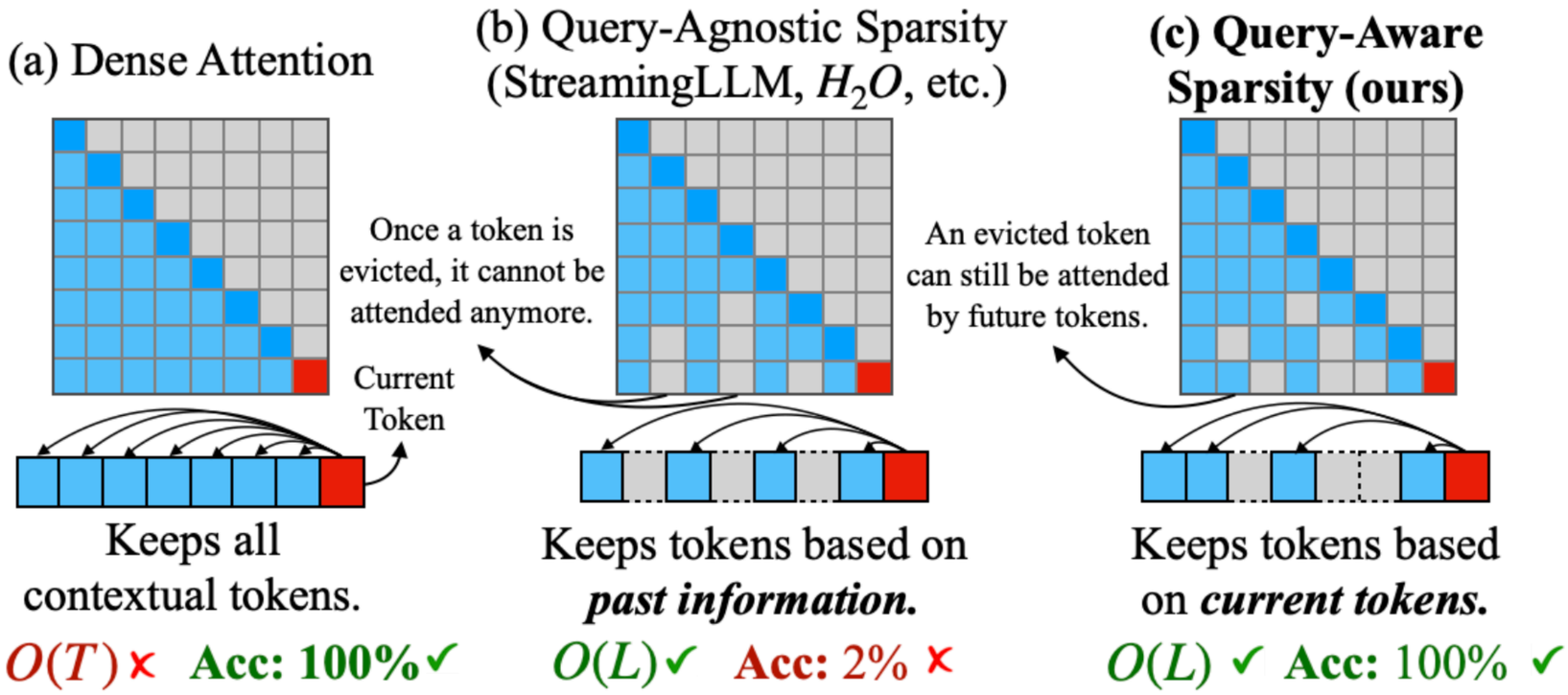
- SpAtten decides which tokens to discard by the sum of their historical attention scores.
- Tokens with small cumulative attention scores are pruned away.



SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. HPCA'21

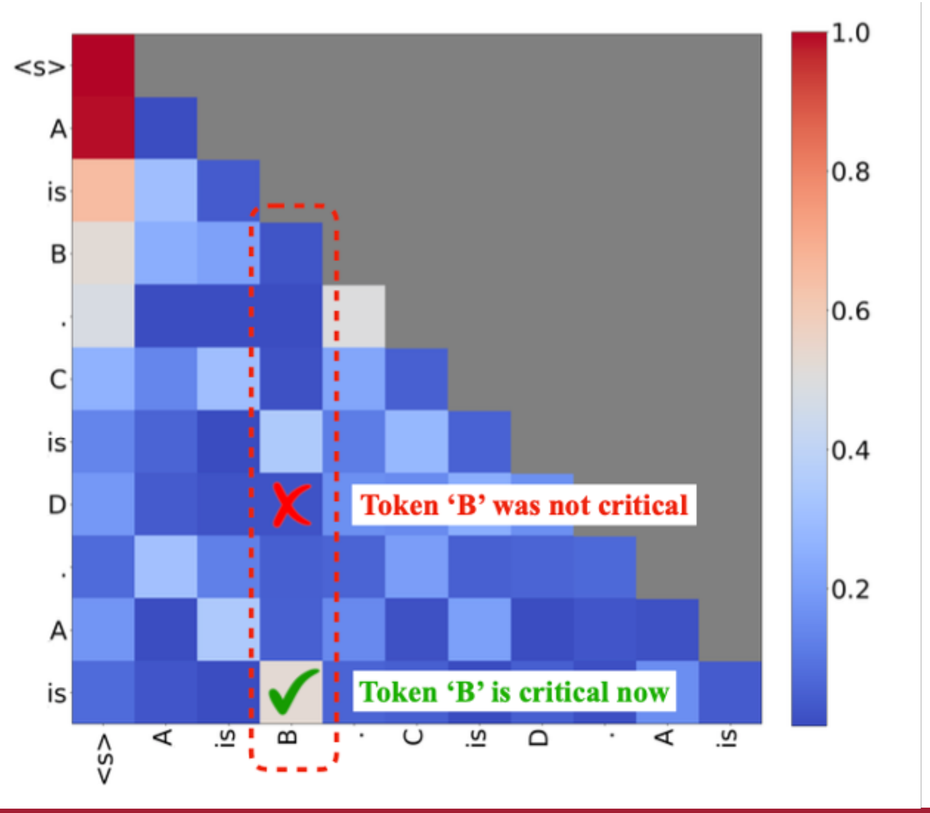
The Limits of Previous Methods

- Many previous efforts have been dedicated to compressing the size of the KV cache to accelerate attention and reduce memory usage.
- These methods decide which parts of the KV cache to discard based on historical information or current states, but **discarded tokens might be important for future tokens**, which may cause the loss of important information.



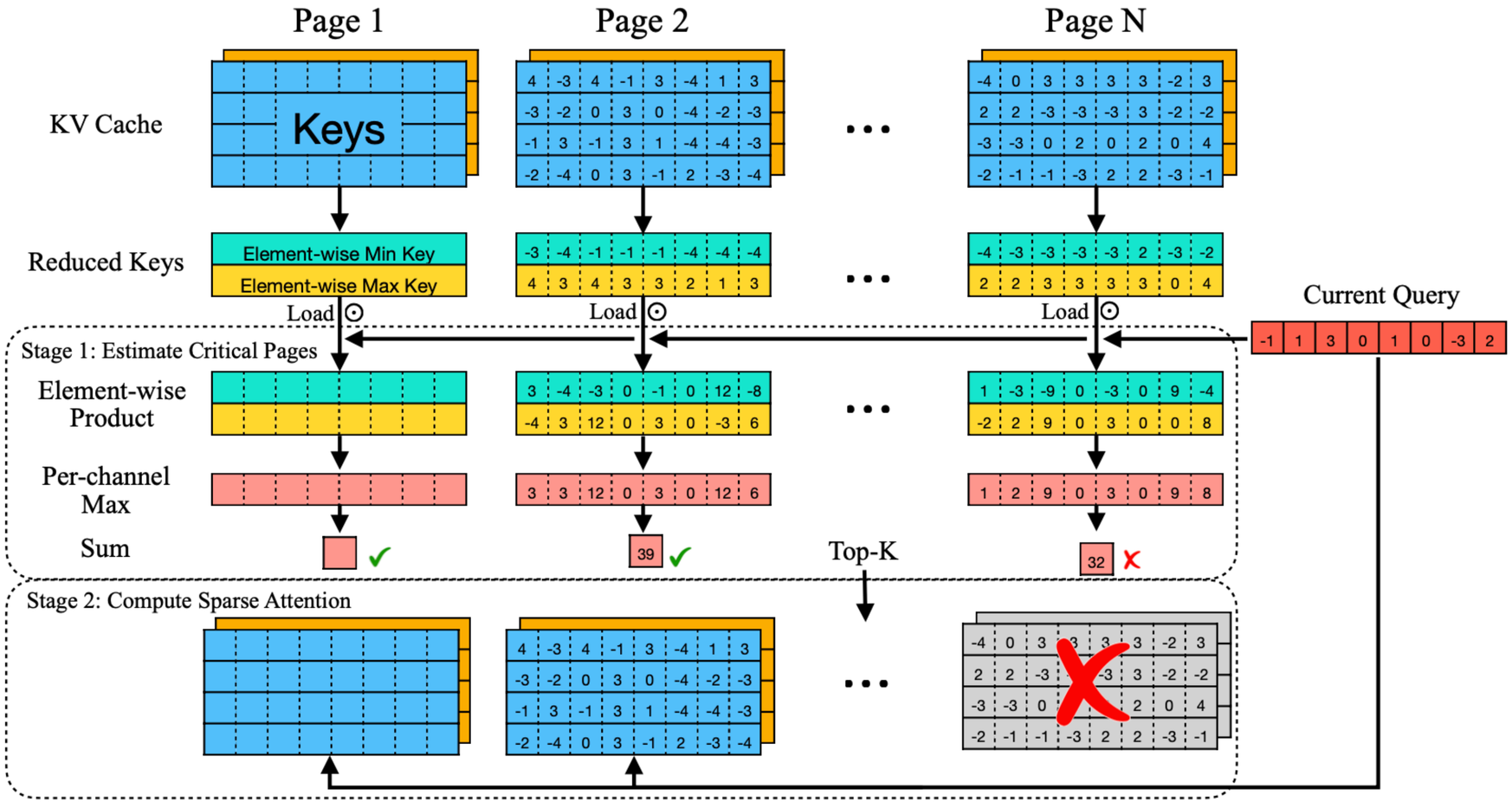
The Limits of Previous Methods

- The **criticality of the tokens is dynamic** and **highly dependent on the query vector Q**.
- Example: the token ‘B’ is critical to the current query ‘is’. Thus, it has a high attention score. However, before the final token ‘is’, ‘B’ is not critical for any previous query and has very low attention scores.



Quest: Using Query-aware Sparsity in Attention

- **Key Idea:** **preserve all KV cache**, and significantly accelerate inference by reducing the memory movement from the entire KV cache to selected constant K pages.



Quest: Using Query-aware Sparsity in Attention

- Our insight is that in order not to miss critical tokens, we should **select pages containing the tokens with the highest attention weights**.
- However, for an efficient selection of pages, we should **calculate an approximate attention score** following this insight.
- We found that the **upper bound attention weights within a page** can be used to approximate the highest attention score in the page.

When inserting new token to KV cache:
Input: Key vector K, Dimension of hidden states dim, Current maximal vector M_i, Current minimal vector m_i

for $i = 1$ **to** dim **do**
  $M_i = \max(M_i, k_i)$
  $m_i = \min(m_i, k_i)$
end for

When perform self-attention:
Input: Query vector Q, Dimension of hidden states dim, Current maximal vector M_i, Current minimal vector m_i

Initialize $score = 0$.
for $i = 1$ **to** dim **do**
  $score$ += $MAX(q_i * max, q_i * min)$
end for

Quest Performance

Needle-in-a-Haystack

- (i) Results of 10k length passkey retrieval test on LongChat-7b-v1.5-32k.
- (ii) Results of 100k length passkey retrieval test on Yarn-Llama-2-7b-128k.
- Quest can achieve nearly **perfect accuracy** with a KV cache of 64 and 1024 tokens, which is about **1% of the total sequence length**, demonstrating that Quest can effectively preserve the model's ability to handle long-dependency tasks.

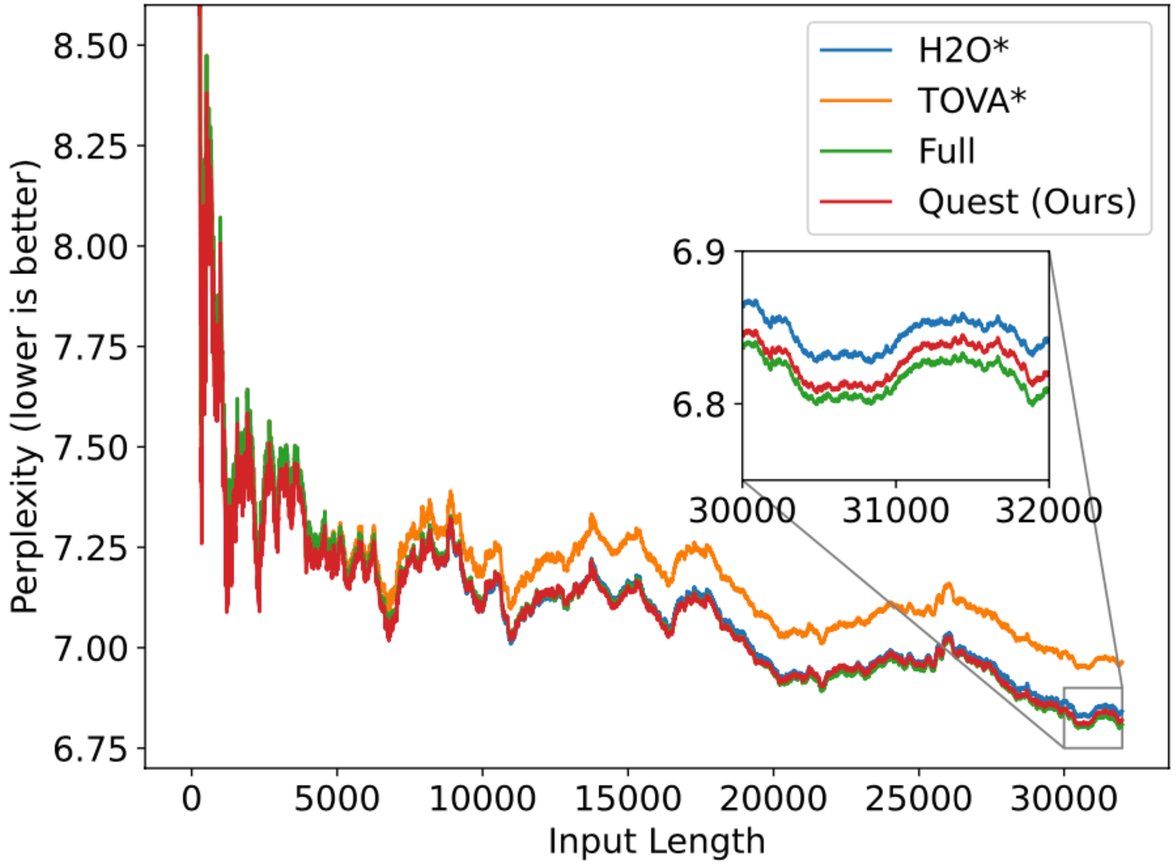
Method / Budget	32	64	128	256	512
H2O	0%	1%	1%	1%	3%
TOVA	0%	1%	1%	3%	8%
StreamingLLM	1%	1%	1%	3%	5%
Quest (ours)	**65%**	**99%**	**99%**	**99%**	**100%**

Method / Budget	256	512	1024	2048	4096
H2O	2%	2%	2%	2%	4%
TOVA	2%	2%	2%	2%	10%
StreamingLLM	1%	1%	1%	2%	4%
Quest (ours)	**88%**	**92%**	**96%**	**100%**	**100%**

Quest Performance

Super Long Language Modeling

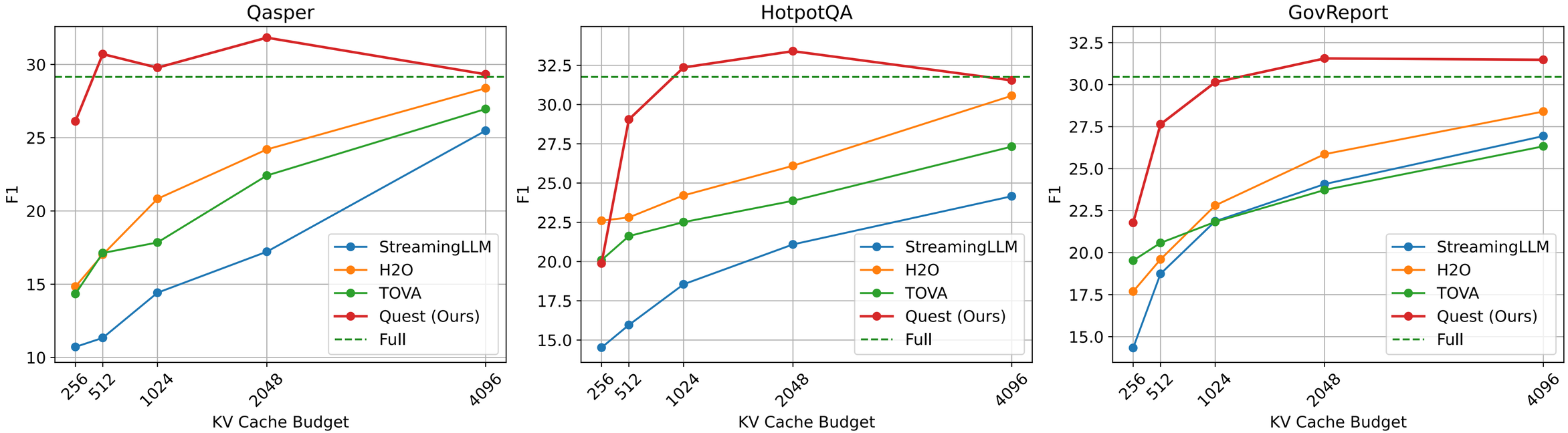
- Language modeling evaluation of Quest on PG19 dataset.
- Quest can **closely match the performance of the full cache model**.



Quest Performance

LongBench

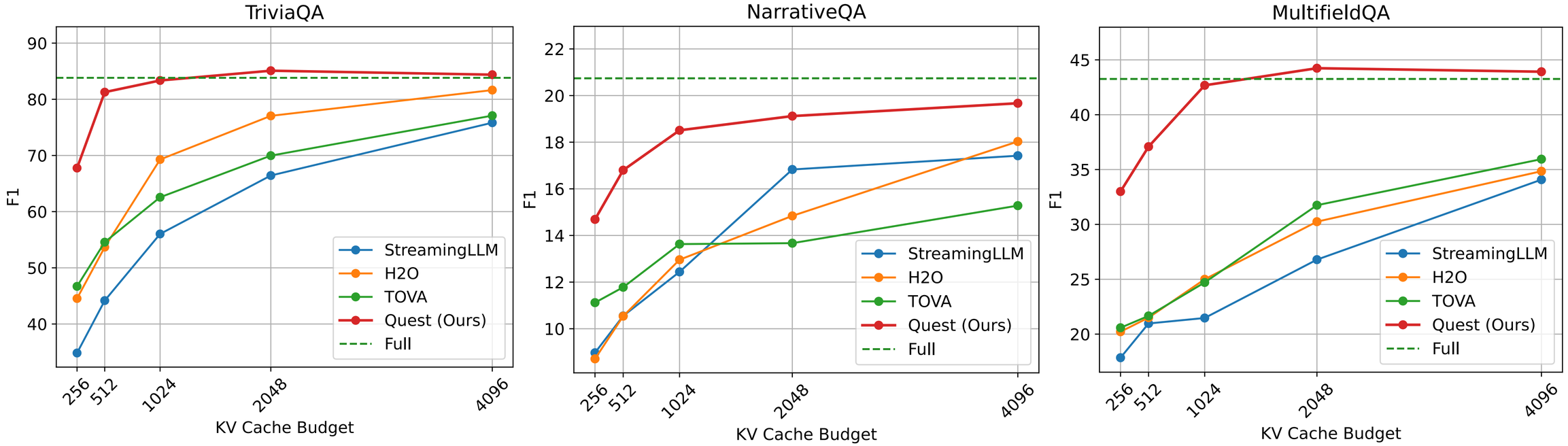
- We evaluate LongChat-7b-v1.5-32k across a wide range of long-context datasets,
- Quest with **a budget of 2k tokens can achieve comparable performance as the model with full KV cache**, while other baselines still exhibit a notable gap from full cache performance even with a larger budget.
- Single-document QA: NarrativeQA, Qasper, MultiFieldQA; multi-document QA: HotpotQA; summarization: GovReport; few-shot learning: TriviaQA.



Quest Performance

LongBench

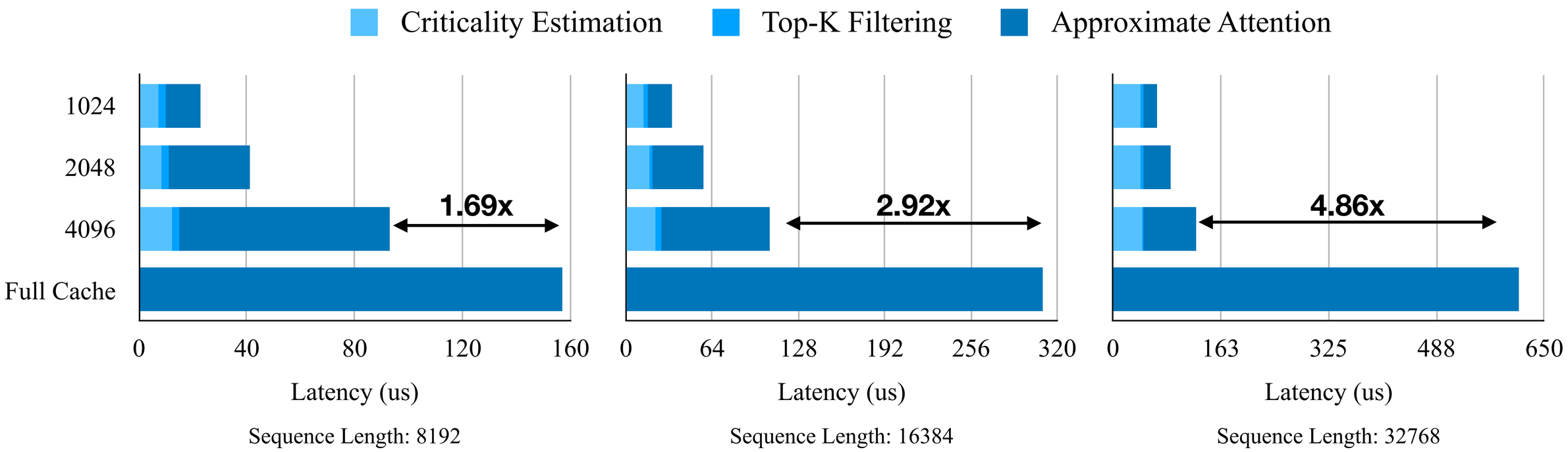
- We evaluate LongChat-7b-v1.5-32k across a wide range of long-context datasets,
- Quest with **a budget of 2k tokens can achieve comparable performance as the model with full KV cache**, while other baselines still exhibit a notable gap from full cache performance even with a larger budget.
- Single-document QA: NarrativeQA, Qasper, MultiFieldQA; multi-document QA: HotpotQA; summarization: GovReport; few-shot learning: TriviaQA.



Efficiency Evaluation

Quest attention time breakdown

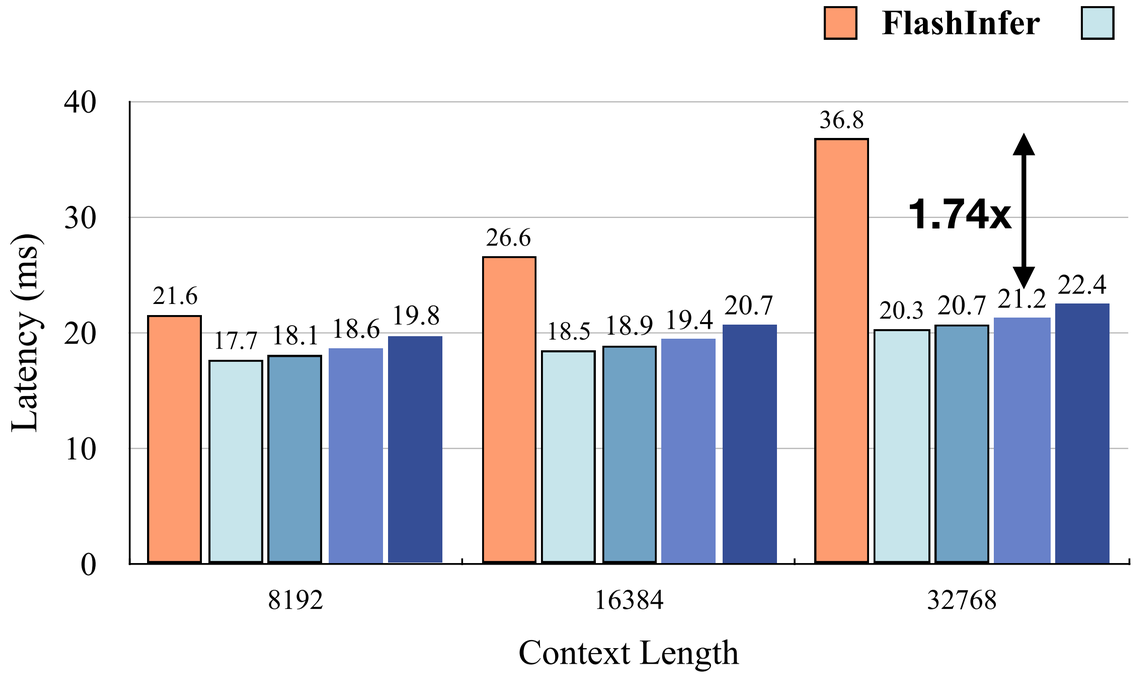
- At all sequence lengths, Quest significantly outperforms FlashInfer, as the memory movement is reduced.
- **Quest speeds up self-attention by 7.03×** at sequence length 32k with token budget 2048.



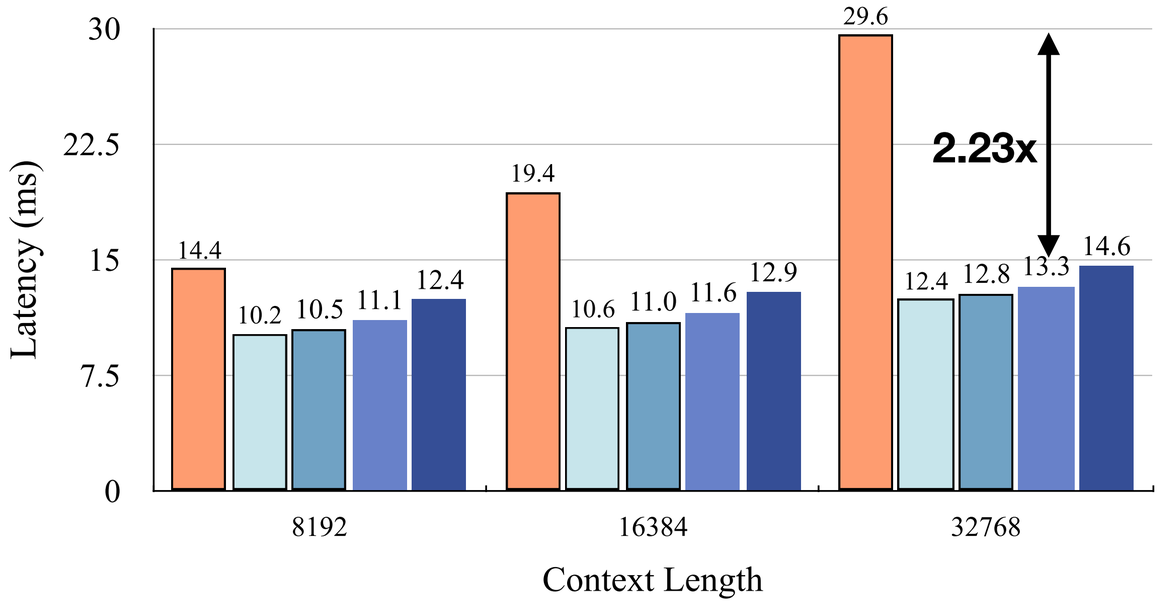
Efficiency Evaluation

End-to-end latency

- For all sequence lengths, Quest significantly outperforms FlashInfer. Increasing the sequence lengths only slightly changes the latency of Quest.
- **Quest speedup end-to-end inference by 2.23×** with sequence length 30K, token budget 2048, 4-bit weight quantization.



(a) FP16 Weight

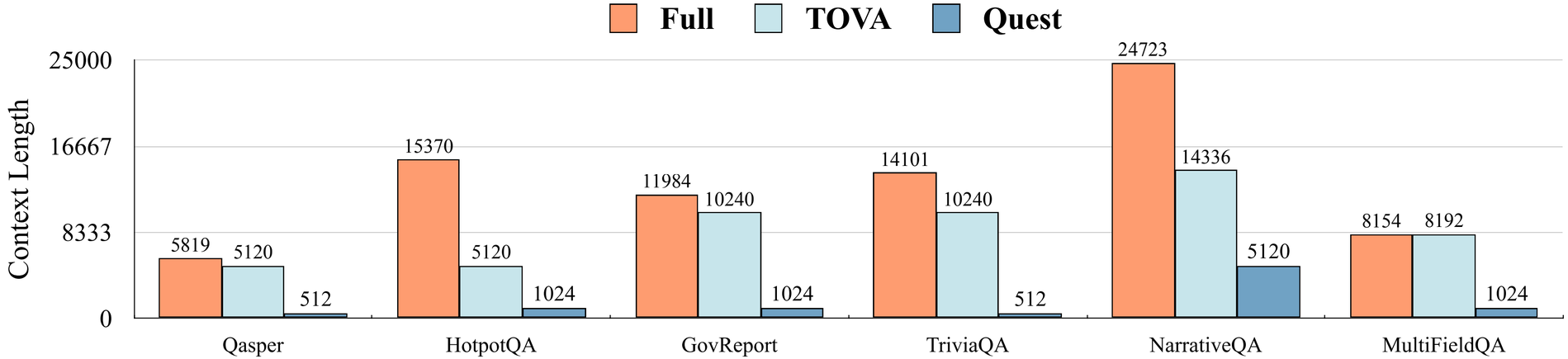


(b) 4-bit Weight (AWQ)

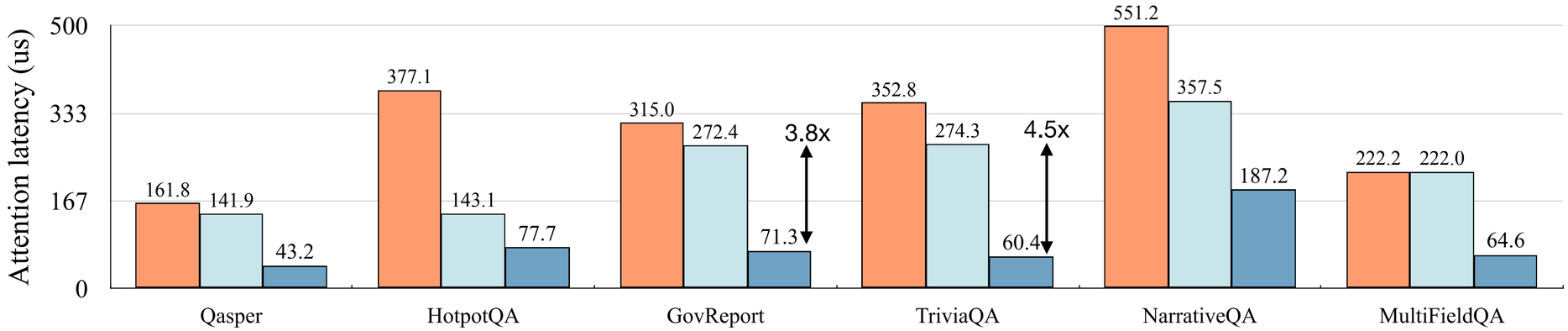
Efficiency Evaluation

Efficiency comparison with baselines

- Baselines need nearly full cache to achieve lossless performance on LongBench benchmarks.
- Therefore, **Quest outperforms the baseline by up to 4.54x with the same lossless accuracy**.



(a) Average context length for lossless accuracy of attention mechanisms

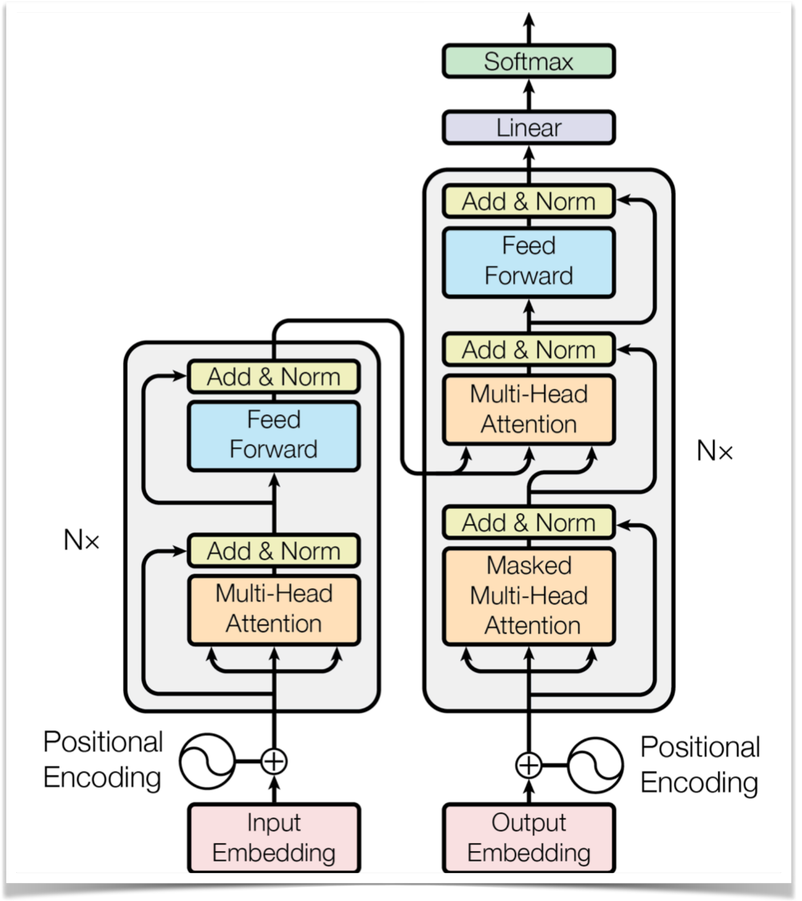


(b) Inference latency of different attention mechanism under benchmarks

Lecture Plan

Today, we will cover:

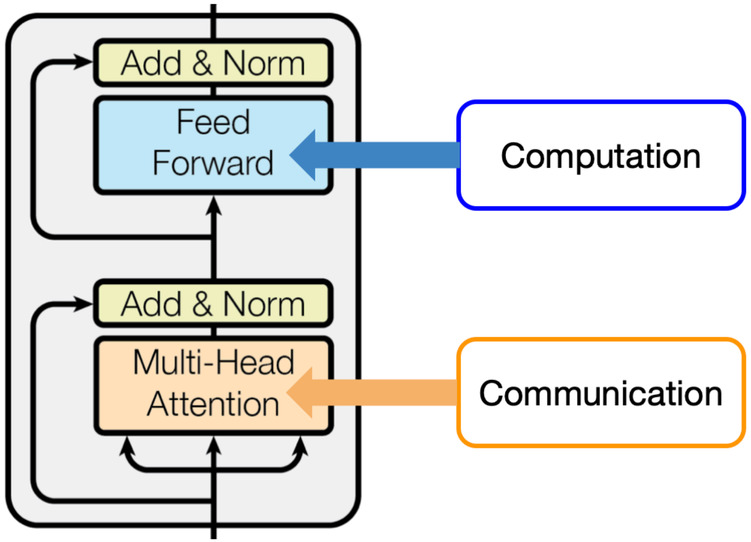
1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. **Beyond Transformers**
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



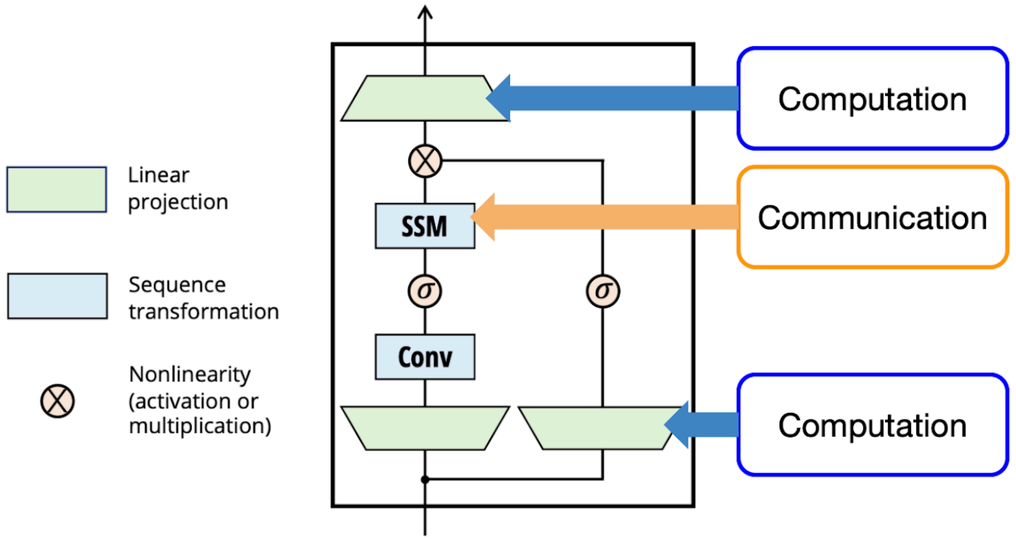
State-Space Models (SSMs): Mamba

Solving the Quadratic Complexity of Transformers in Handling Long Sequences

- Two important operations in LLMs:
  - **Communication** *between* tokens (Transformers: **Attention =>** Mamba: **SSM**)
  - **Computation** *within* a token (MLPs)
- **Core Idea:** Mamba substitutes Attention with **State Space Models (SSMs)** for handling sequence communication, achieving **linear-time processing** for long sequences.



Transformer Block



Mamba Block

Mamba: Linear-Time Sequence Modeling with Selective State Spaces [Gu and Dao, 2024]
https://thegradient.pub/mamba-explained/

State-Space Models (SSMs): Mamba

What is a State Space Model (SSM)?

- **State (h):** Represents the current knowledge about the sequence.
- **Input (x):** New information entering the sequence.
- **Update:** The state adjusts dynamically based on input, focusing only on relevant data.

Linear Recurrence

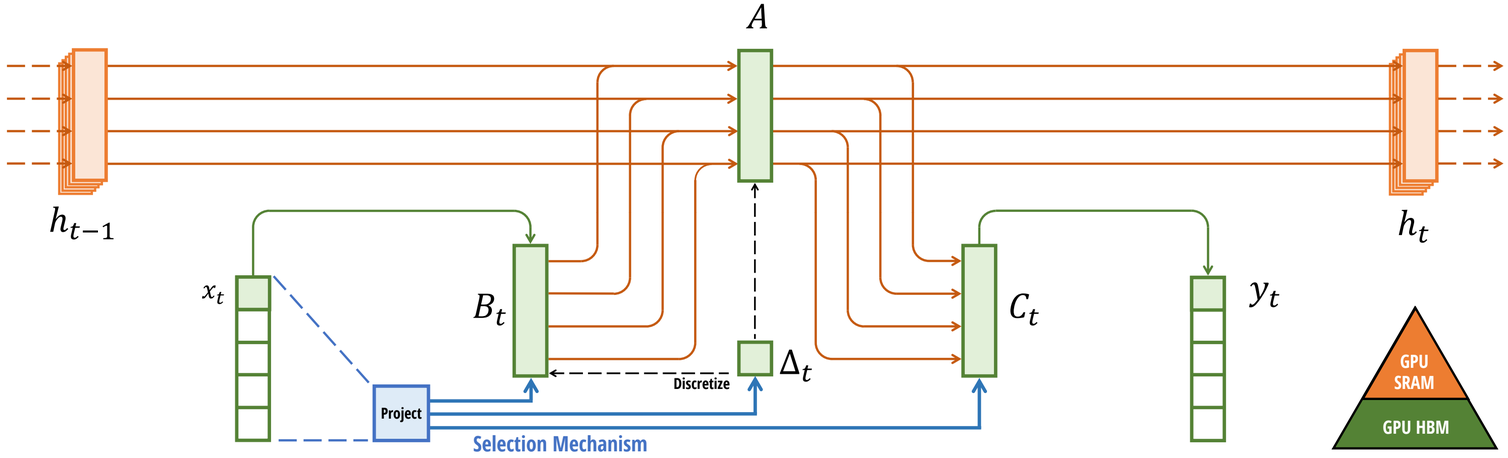
$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

$=$

Global Convolution

$$\overline{K} = (C\overline{B}, C\overline{A}\overline{B}, \ldots, C\overline{A}^k\overline{B}, \ldots)$$
$$y = x * \overline{K}$$

- **A controls state transition:** *How should I forget or update the state over time?*
- **B maps new input to output:** *What part of the new input should I remember?*
- ***C* maps state to output:** *How can I use the state for a good prediction?*



Mamba: Linear-Time Sequence Modeling with Selective State Spaces [Gu and Dao, 2024]

State-Space Models (SSMs): Mamba

The Selection Mechanism: SSM -> Mamba (*Selective* State Space Model)

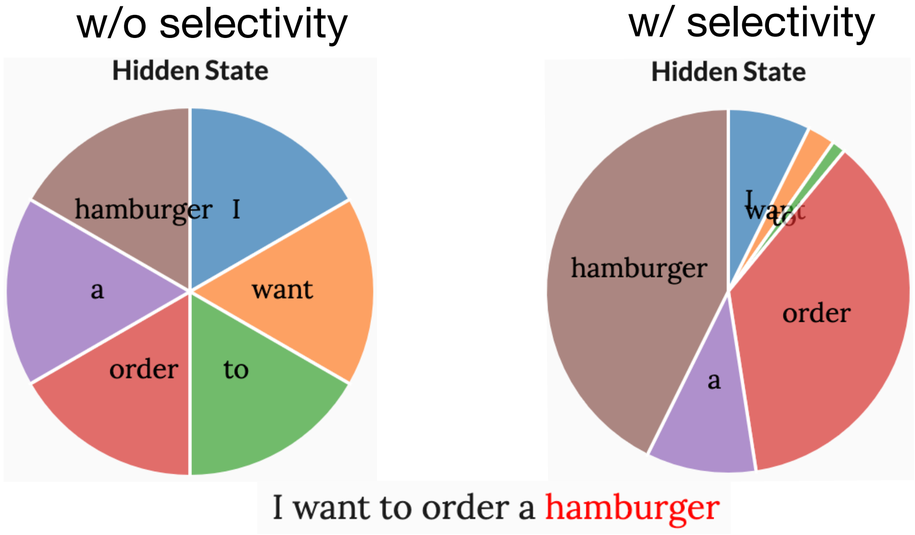
- **Selectivity** allows each token to be transformed into the state in a way that is unique to its own needs. The *transformation matrices (A, B, and C)* become **input (x) dependent.**
  - **In regular SSMs, A, B, and C are input independent learned matrices.**

$$\mathbf{A} = \mathbf{A}_\theta \text{ etc. (where } \theta \text{ represents the learned parameters)}$$

  - **In Mamba, A, B, and C are also functions of the input (x)**

$$\mathbf{A} = \mathbf{A}_{\theta(x)} \text{ etc; the matrices are context dependent rather than static.}$$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$



Mamba: Linear-Time Sequence Modeling with Selective State Spaces [Gu and Dao, 2024]

State-Space Models (SSMs): Mamba

The Problem with Selectivity: Convolution Cannot Be Used for Training

- When there is no **selectivity,** we can speed up training with **convolution**, i.e., precompute the left-hand matrices for all inputs and save them as **convolutional kernel K**.

$$
\begin{aligned}
h_0 &= \bar{\mathbf{B}}x_0 \\
h_1 &= \bar{\mathbf{A}}(\bar{\mathbf{B}}x_0) + \bar{\mathbf{B}}x_1 \\
h_2 &= \bar{\mathbf{A}}(\bar{\mathbf{A}}(\bar{\mathbf{B}}x_0) + \bar{\mathbf{B}}x_1) + \bar{\mathbf{B}}x_2 \\
h_3 &= \bar{\mathbf{A}}(\bar{\mathbf{A}}(\bar{\mathbf{A}}(\bar{\mathbf{B}}x_0) + \bar{\mathbf{B}}x_1) + \bar{\mathbf{B}}x_2) + \bar{\mathbf{B}}x_3
\end{aligned}
$$

$$
\begin{aligned}
y_3 &= \mathbf{C}(\bar{\mathbf{A}}(\bar{\mathbf{A}}(\bar{\mathbf{A}}(\bar{\mathbf{B}}x_0) + \bar{\mathbf{B}}x_1) + \bar{\mathbf{B}}x_2) + \bar{\mathbf{B}}x_3) \\
y_3 &= \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{B}}x_0 + \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{B}}x_1 + \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}x_2 + \mathbf{C}\bar{\mathbf{B}}x_3
\end{aligned}
$$

$$
y_3 = \begin{pmatrix} \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{B}} & \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{A}}\bar{\mathbf{B}} & \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}} & \mathbf{C}\bar{\mathbf{B}} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}
$$

Linear Recurrence

$$
\begin{aligned}
h_t &= \overline{A}h_{t-1} + \overline{B}x_t \\
y_t &= Ch_t
\end{aligned}
$$

$=$

Global Convolution

$$
\begin{aligned}
\overline{K} &= (C\overline{B}, C\overline{AB}, \ldots, C\overline{A}^k\overline{B}, \ldots) \\
y &= x * \overline{K}
\end{aligned}
$$

- However, selectivity presents us with a problem: **A, B, and C changes depending on the input!**

$\mathbf{A} = \mathbf{A}_{\theta(x)}$ etc; the matrices are context dependent rather than static.

~~$y = \bar{\mathbf{K}} * x$~~

- Training with the **linear recurrence form (RNN mode)** will be very slow, can we accelerate it?

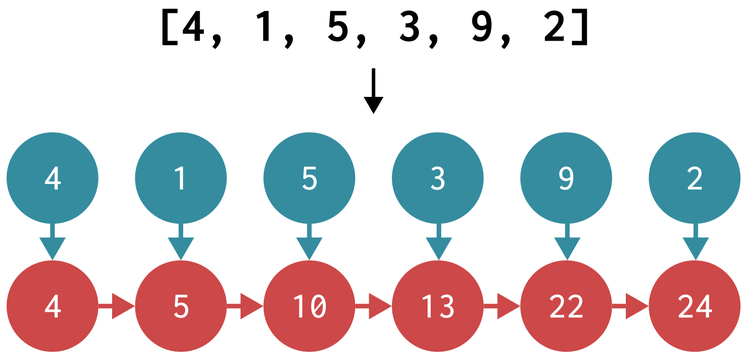
Mamba: Linear-Time Sequence Modeling with Selective State Spaces [Gu and Dao, 2024]

State-Space Models (SSMs): Mamba

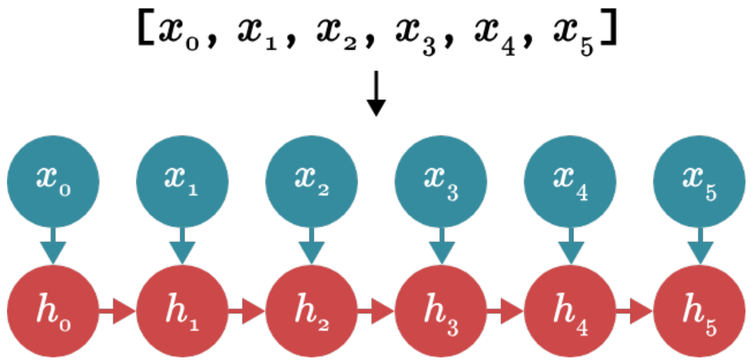
Fast Training without Convolution: Using Parallel Scan

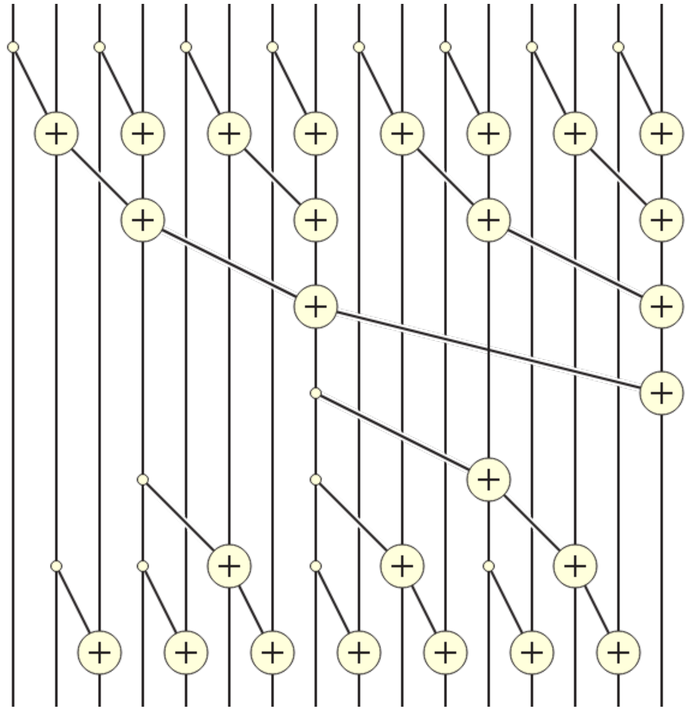
- The process of **computing hidden states of Mamba** is very similar to **computing the prefix sum of an array:** we can use the **parallel scan algorithm** to accelerate!

Computing the prefix sum of an array



Computing hidden states of Mamba





Parallel scan Algorithm

Mamba: Linear-Time Sequence Modeling with Selective State Spaces [Gu and Dao, 2024]
Scans as primitive parallel operations [Blelloch, 1989]

Hybrid Models: Jamba

A Hybrid Transformer-Mamba Language Model

- Combines **Transformer** and **Mamba** layers with Mixture-of-Experts (MoE) modules.
- Balances memory efficiency, high throughput, and high performance.
- **Innovative Model Design:**
  - Interleaves Transformer and Mamba layers to reduce memory demands.
  - MoE layers increase model capacity while keeping active parameters low.
  - Fits into a single 80GB GPU with support for 256K tokens.

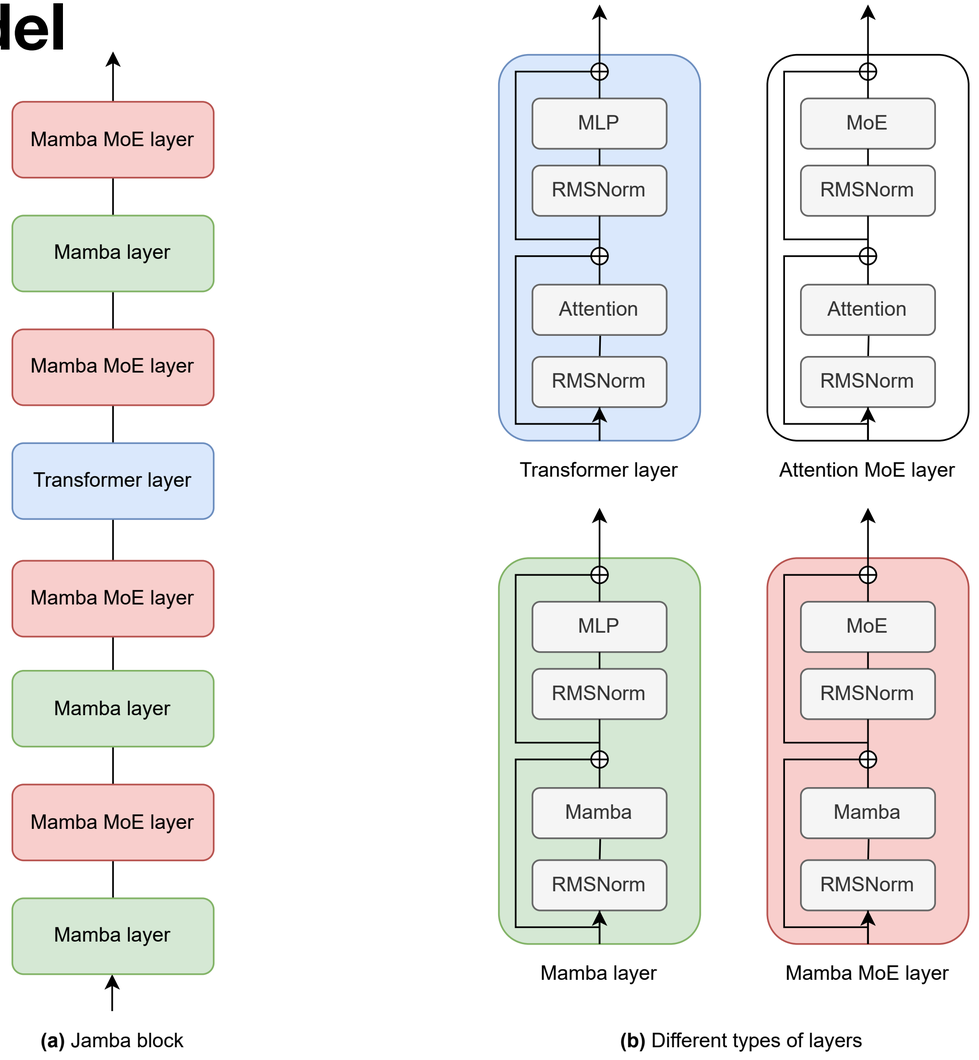
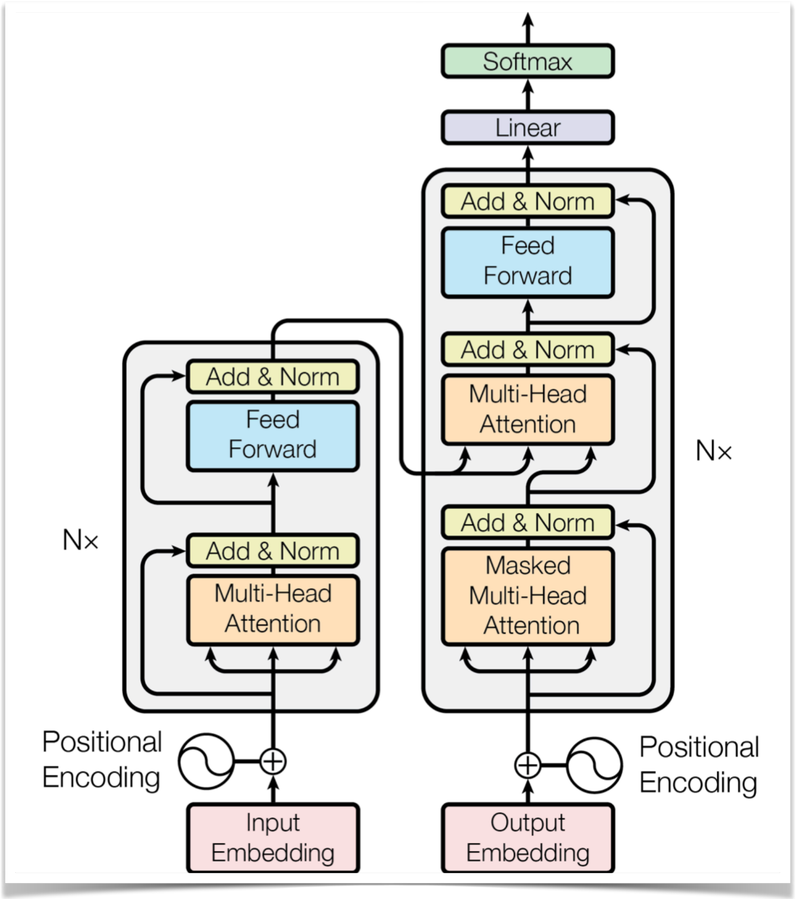


Figure 1: **(a)** A single Jamba block. **(b)** Different types of layers. The implementation shown here is with $l = 8$, $a : m = 1 : 7$ ratio of attention-to-Mamba layers, and MoE applied every $e = 2$ layers.

Jamba: A Hybrid Transformer-Mamba Language Model [Lieber et al., 2024]

Summary of Today's Lecture

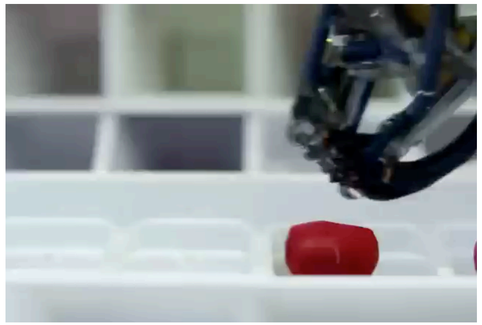
1. **Context Extension**
    1. Recap: Rotary Position Embedding (RoPE)
    2. LongLoRA
2. **Evaluation of Long-Context LLMs**
    1. The Lost-in-the-Middle Phenomenon
    2. Long-Context Benchmarks: NIAH, LongBench
3. **Efficient Attention Mechanisms**
    1. Recap: KV Cache
    2. StreamingLLM and Attention Sinks
    3. DuoAttention: Retrieval Heads and Streaming Heads
    4. Quest: Query-Aware Sparsity
4. **Beyond Transformers**
    1. State-Space Models (SSMs): Mamba
    2. Hybrid Models: Jamba



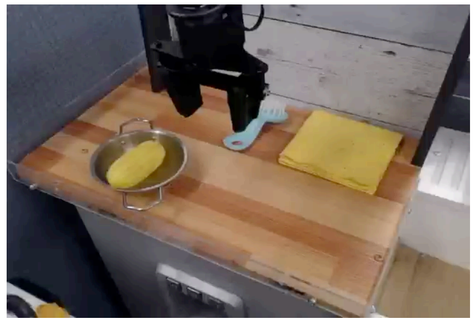
New research needs your help: WorldModelBench

- Question: Can leading video generation videos really create a realistic world?
- Observation: …At least they often violate basic physics laws.

Move without external force



Objects deform and distort



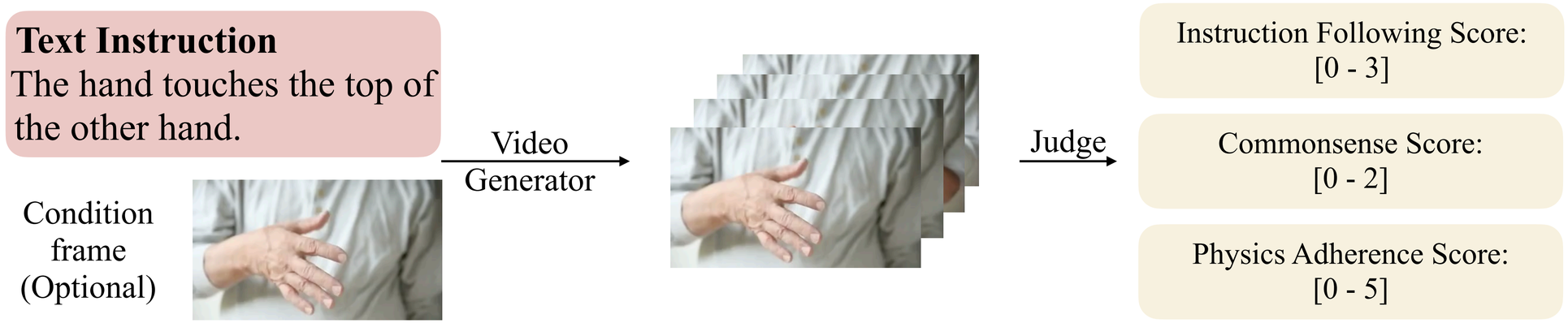
Liquids flow unnaturally

Objects penetrate each others

Violation of gravity

They often also do not follow instruction, and create content that obviously deviate from commonsense.

Example workflow of WorldModelBench



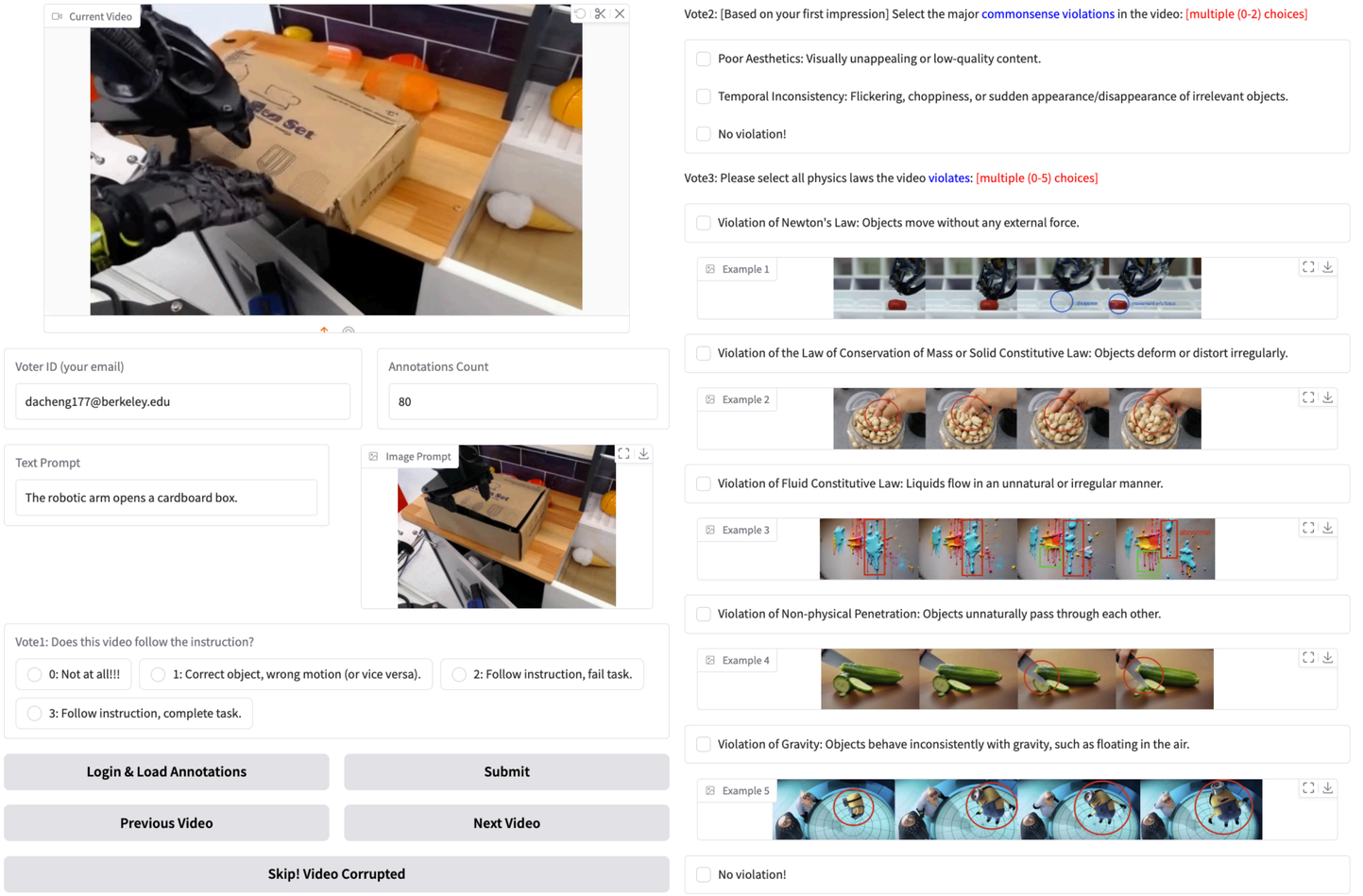
- One sample of our benchmark consists of a text instruction, and a conditional first frame.
- A model generates video based the text instruction, and optionally the condition image (if it takes image condition).
- The benchmark judges the generated video and output a score of 0-15.

Goto: https://worldmodelbench.hanlab.ai
and help us make some labels!

We need your help!

Goto: https://worldmodelbench.hanlab.ai
and help us make some labels!

- Existing automatic judgement (e.g. VLM-as-a-judge) is not reliable towards the goal. We need golden human label.



Competition for annotations

Bonus: Students that annotate most will get rewarded (against quality check)!

- 1st: Jeston Orin Nano
- 2nd-3rd: Qualcomm Snapdragon Development Board
- 4nd-6nd: STM32 TinyML Development board

Instructions:

https://docs.google.com/document/d/1kOirJAAU-g7CwrI7NtfJy6hYsiU0VChGJivr7_-b860/edit?usp=sharing

Website: https://worldmodelbench.hanlab.ai

Goto: https://worldmodelbench.hanlab.ai
and help us make some labels!

Competition start date: 2024.10.29 5 PM ET

Competition end date: 2024.11.04 11:59:59 PM ET (Next Monday)

Bonus release: Next Tuesday in class.

Note: **Please use your mit email address.**