KV Cache Compression

Gabriel Pernell, Alexander Martin

JOHNS HOPKINS UNIVERSITY

Background & Motivation

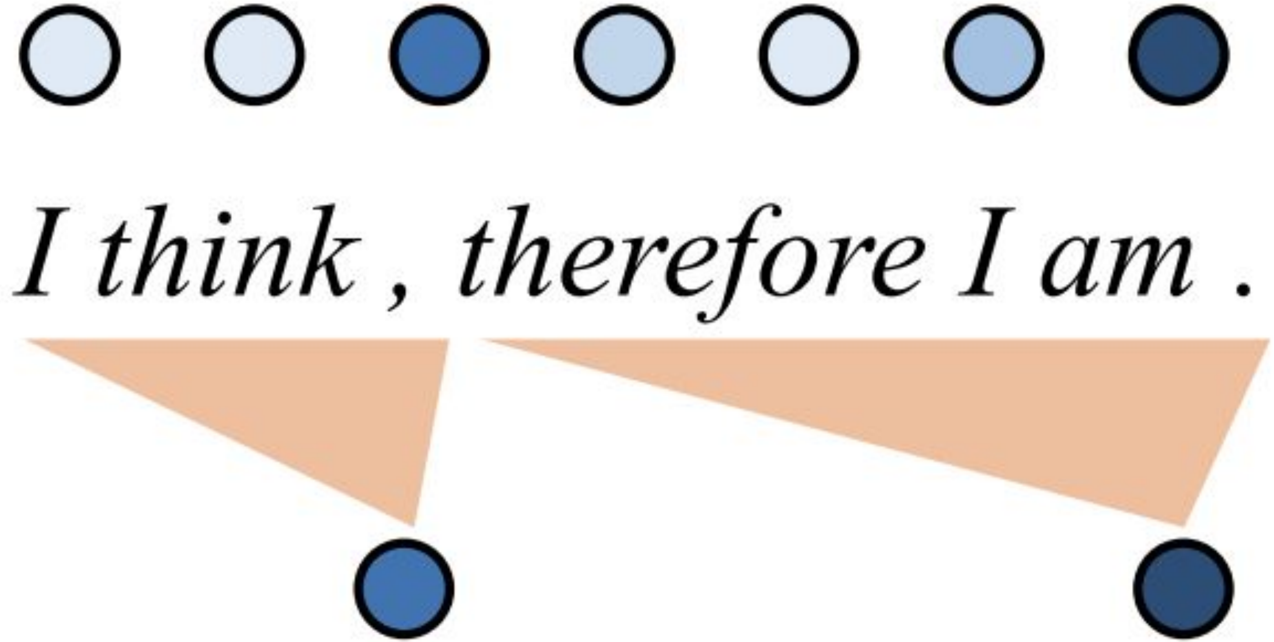
Prefix Caching (Prefilling)

https://blog.squeezebits.com/vllm-vs-tensorrtllm-12-automatic-prefix-caching-38189

SqueezeBits

Token Selection / Eviction

Tokens

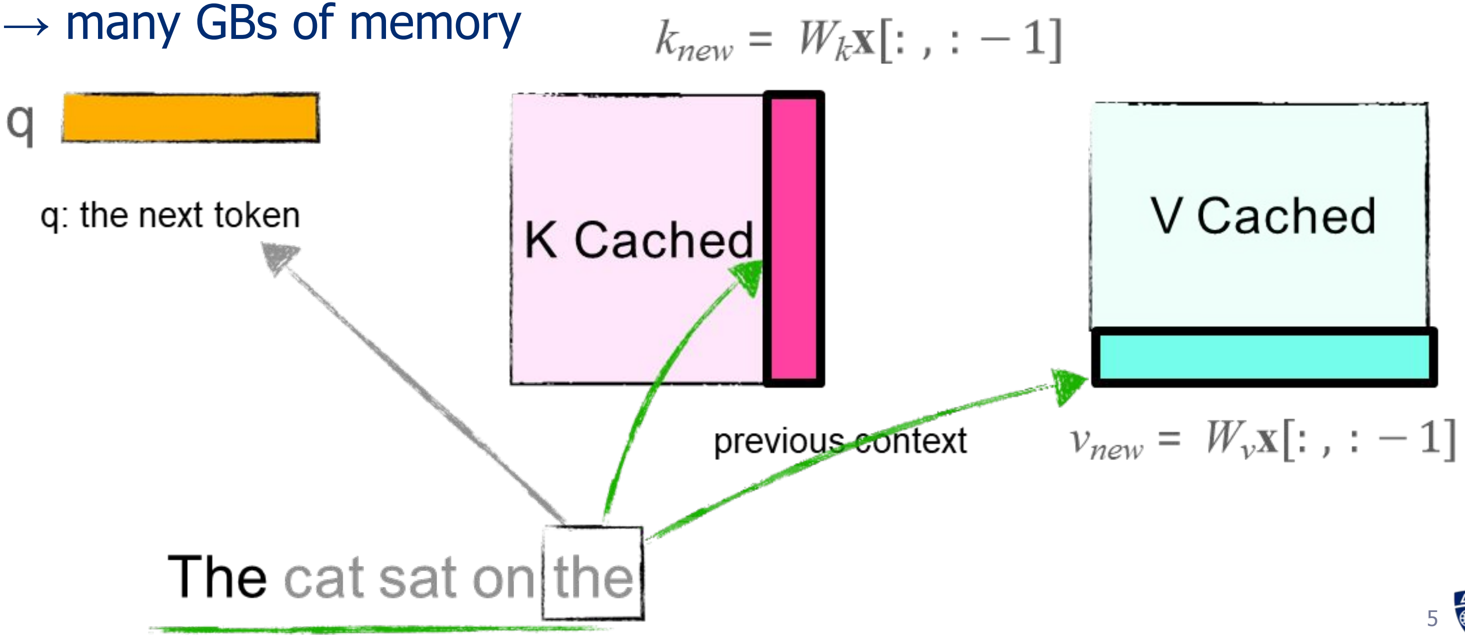


Selected
Tokens

Reminder: Longer Context = Longer Inference Time

Long inference times motivate using a KV Cache

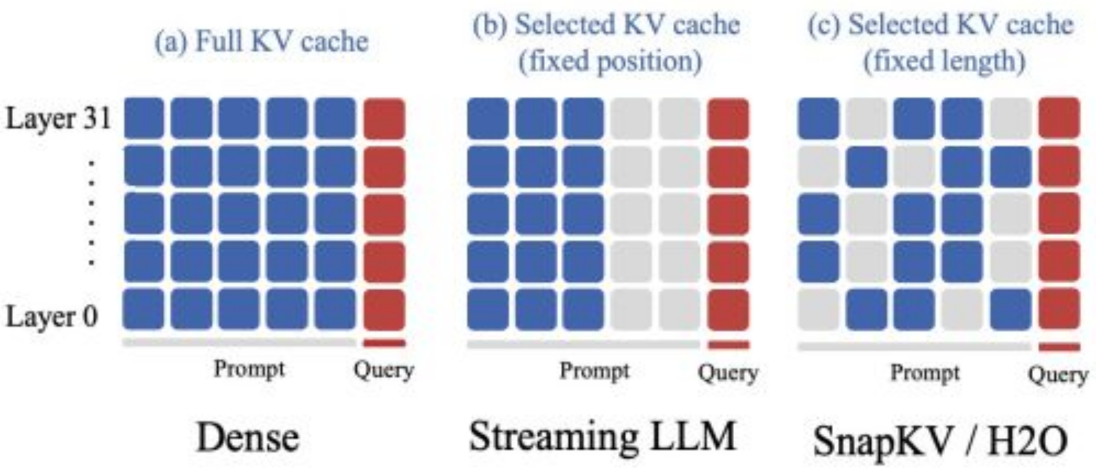
- Cache the keys and values to reduce redundancy and save inference time
- Memory required: $2bndLk$
- Larger models → many GBs of memory



KV Cache Compression

KV cache memory consumption with larger models motivates cache compression & optimization methods.

- Existing methods use a fixed cache size per transformer layer
  - Is this efficient?
  - Authors say no, does not take into account attention varying by layer.



Goals & Research Questions

- Do LLMs aggregate information in recognizable patterns across layers?
  - If so, can this inform a smarter KV cache compression method?
- Goal: develop a compression method that
  - Allocates KV cache size based on layer and attention patterns
  - Preserve long-context performance while reducing memory

PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling

Zefan Cai , Yichi Zhang , Bofei Gao , Yuliang Liu , Yucheng Li , Tianyu Liu , Keming Lu , Wayne Xiong , Yue Dong , Junjie Hu , Wen Xiao

Presented by Gabriel Pernell

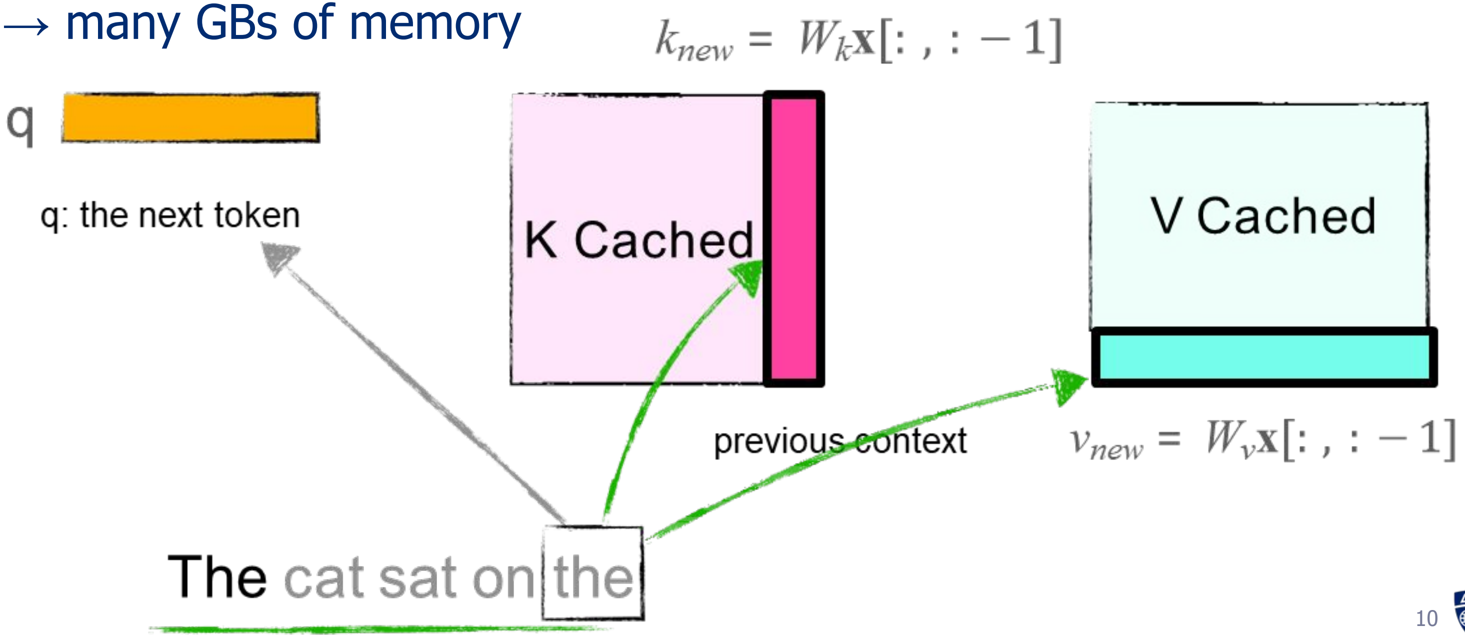
JOHNS HOPKINS UNIVERSITY

Background & Motivation

Reminder: Longer Context = Longer Inference Time

Long inference times motivate using a KV Cache

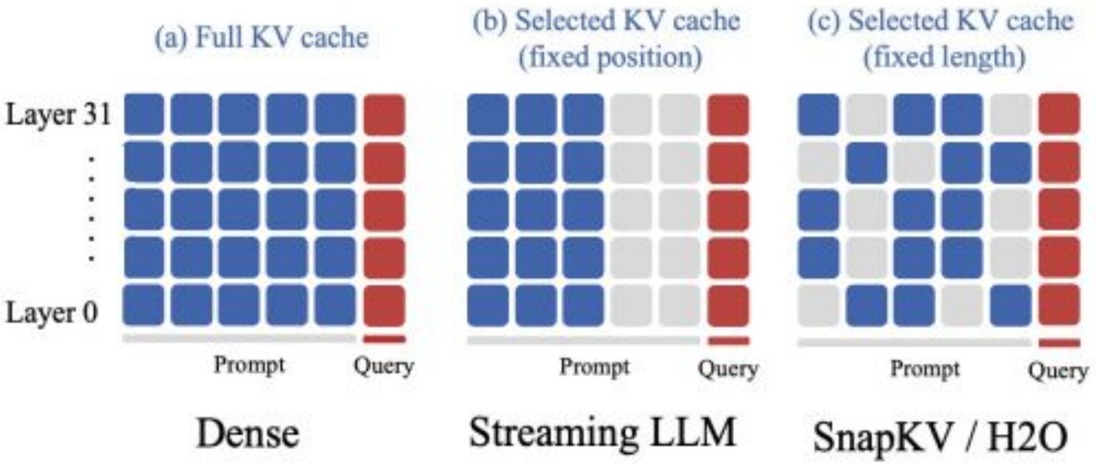
- Cache the keys and values to reduce redundancy and save inference time
- Memory required: $2bndLk$
- Larger models $\rightarrow$ many GBs of memory



KV Cache Compression

KV cache memory consumption with larger models motivates cache compression & optimization methods.

- Existing methods use a fixed cache size per transformer layer
  - Is this efficient?
  - Authors say no, does not take into account attention varying by layer.



Goals & Research Questions

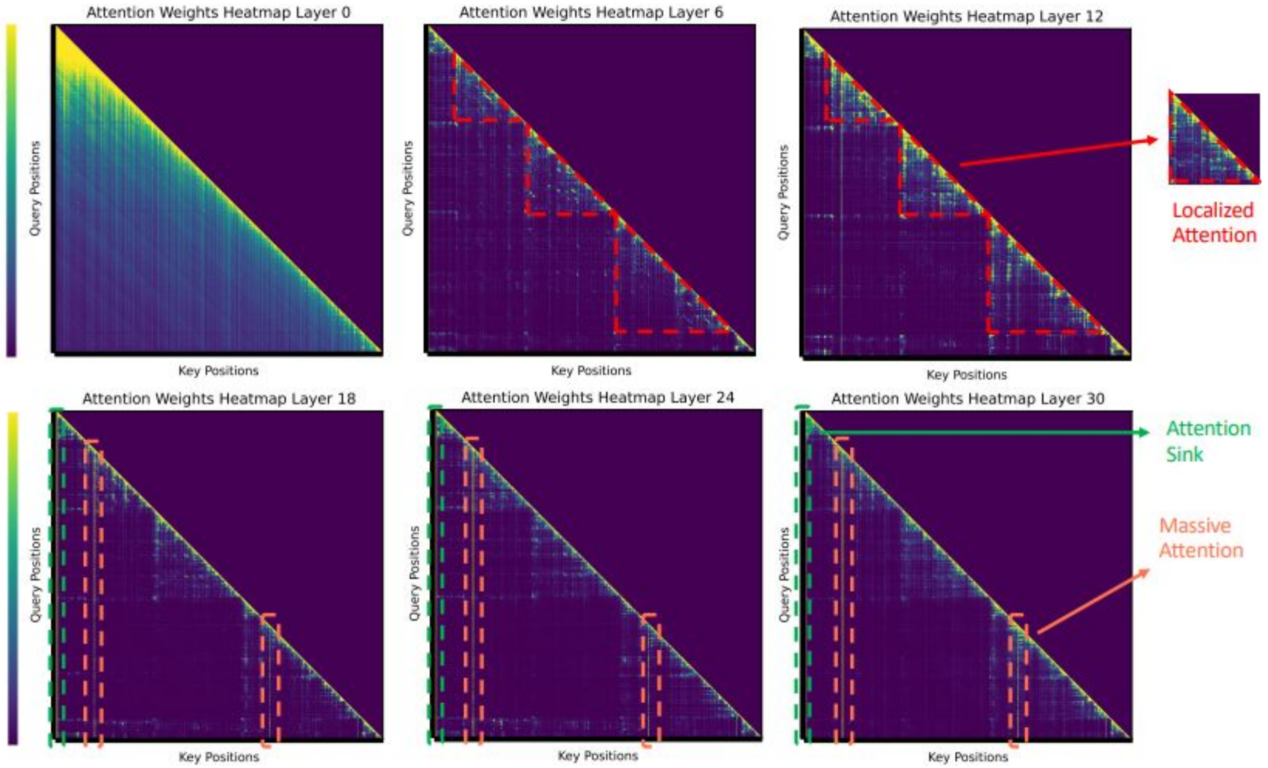
- Do LLMs aggregate information in recognizable patterns across layers?
  - If so, can this inform a smarter KV cache compression method?
- Goal: develop a compression method that
  - Allocates KV cache size based on layer and attention patterns
  - Preserve long-context performance while reducing memory

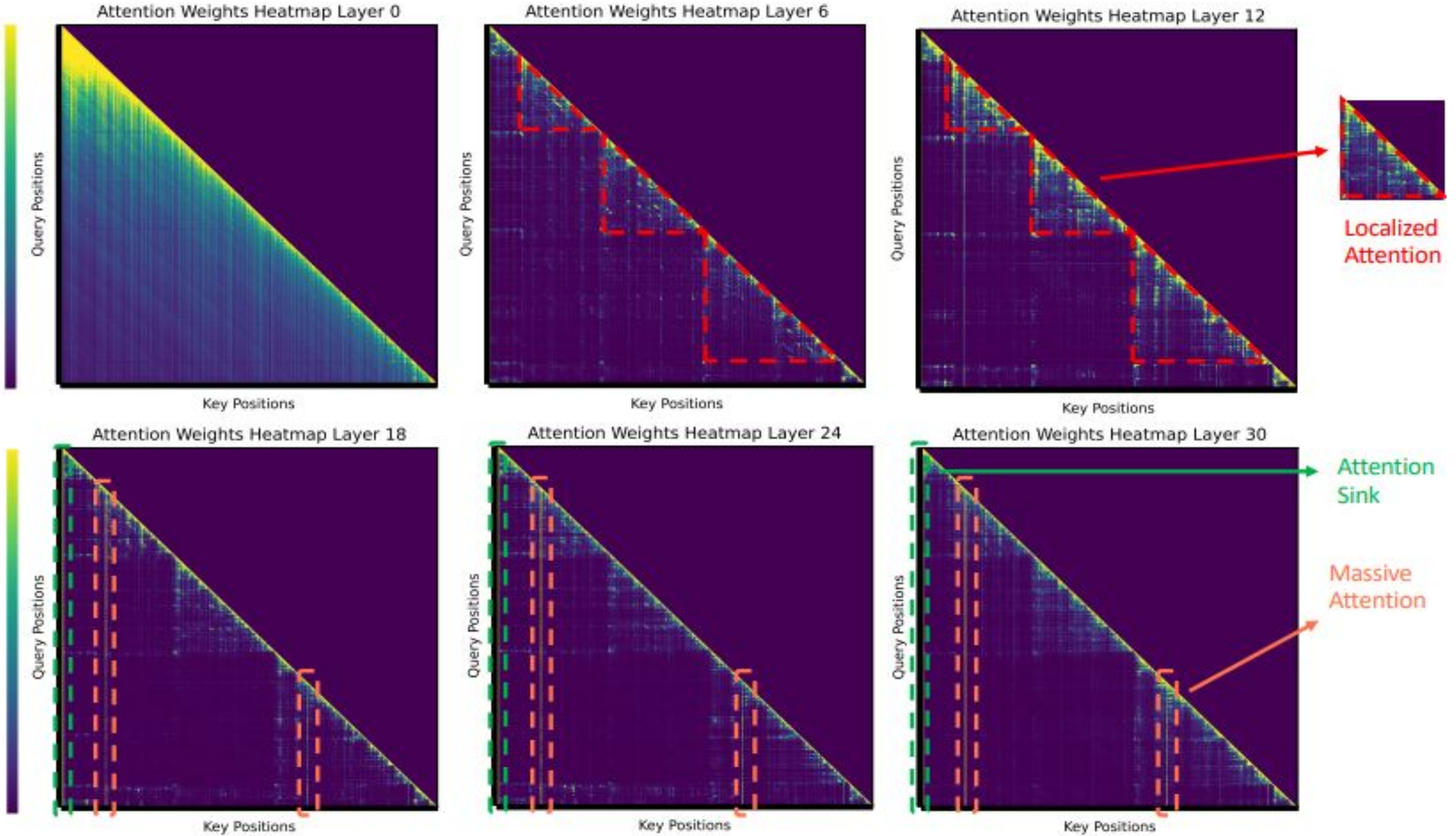
Pyramidal Information Funneling

Observational Study: Information Flow via Attention

Setup: Multi-document QA task on LLaMA: visualized attention scores across layers

- Model given several interrelated documents plus a question.

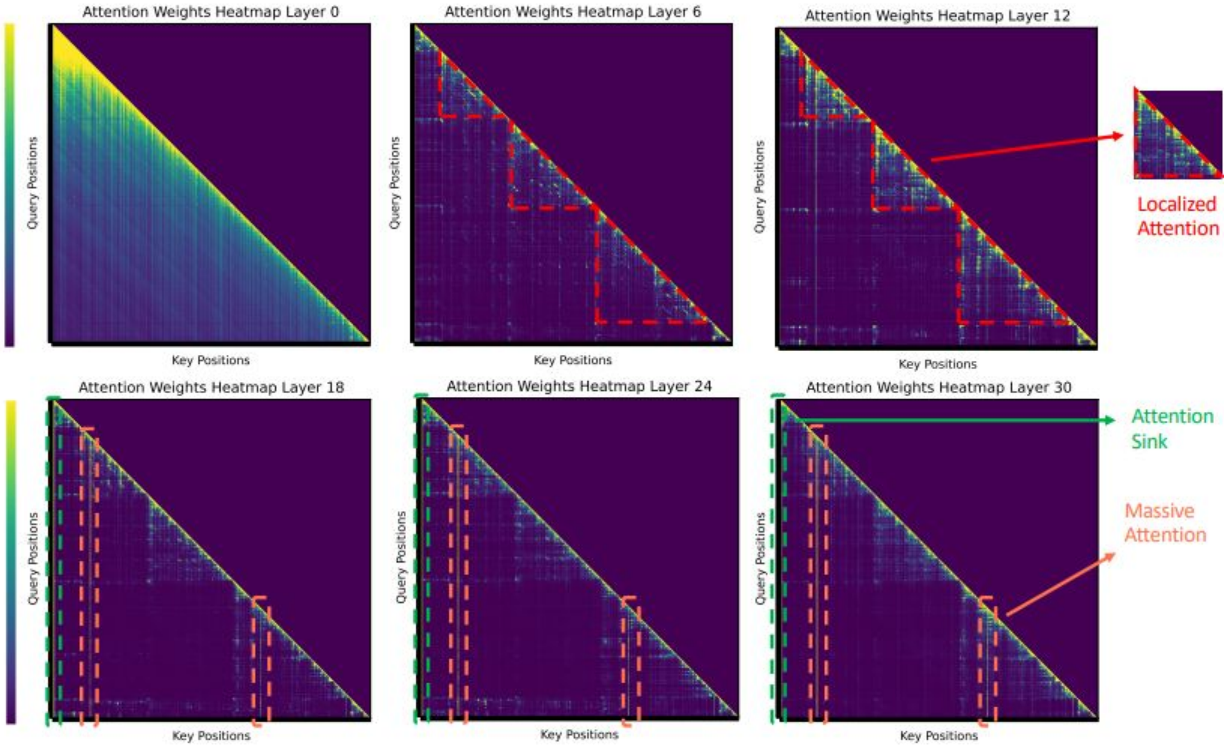




Attention Weights Heatmap Layer 0
Attention Weights Heatmap Layer 6
Attention Weights Heatmap Layer 12
Localized Attention
Attention Weights Heatmap Layer 18
Attention Weights Heatmap Layer 24
Attention Weights Heatmap Layer 30
Attention Sink
Massive Attention
Query Positions
Key Positions

Findings & Interpretation

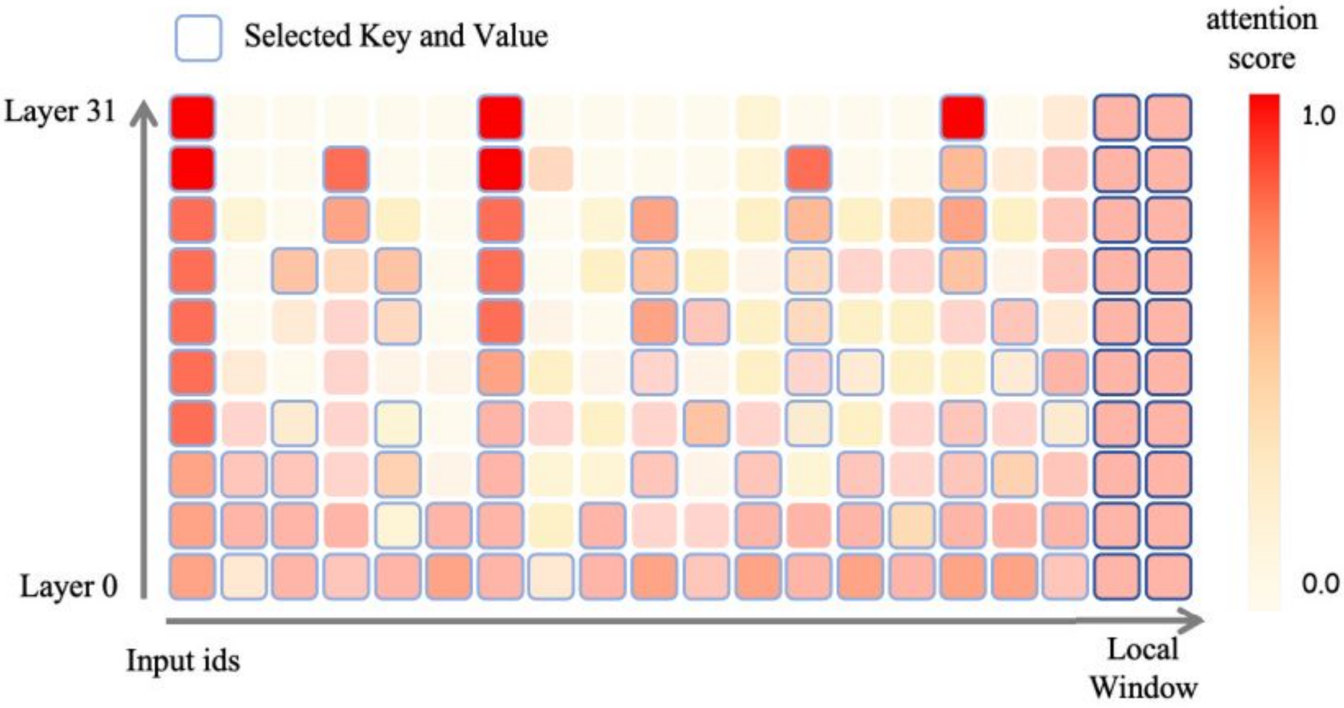
- **Findings:**
  - **Lower Layers:** broad, uniform attention
  - **Middle Layers:** localized attention (e.g. info within documents)
  - **Upper layers:** "massive attention" / attention sink, focus on a few key tokens
- Attention narrows like a pyramid



PyramidKV

PyramidKV: Pyramidal KV Cache Compression Method

More cache is allocated at lower levels, less at higher levels.



Two Key Components (1)

- Dynamic KV Cache Budget Allocation:
  - Retain the KV cache for the last α tokens (instruction tokens)
  - Determine the top and bottom layer budgets:
  - Use an arithmetic sequence to compute cache sizes in between, forming the pyramidal shape.

Total cache budget: $k^{\text{total}} = \sum_{l \in [0,m-1]} k^l$

Top: $k^{m-1} = k^{\text{total}} / (\beta \cdot m)$

Bottom: $k^0 = (2 \cdot k^{\text{total}}) / m - k^{m-1}$

$$k^l = k^0 - \frac{k^0 - k^{m-1}}{m-1} \times l.$$

Two Key Components (2)

KV Cache Selection

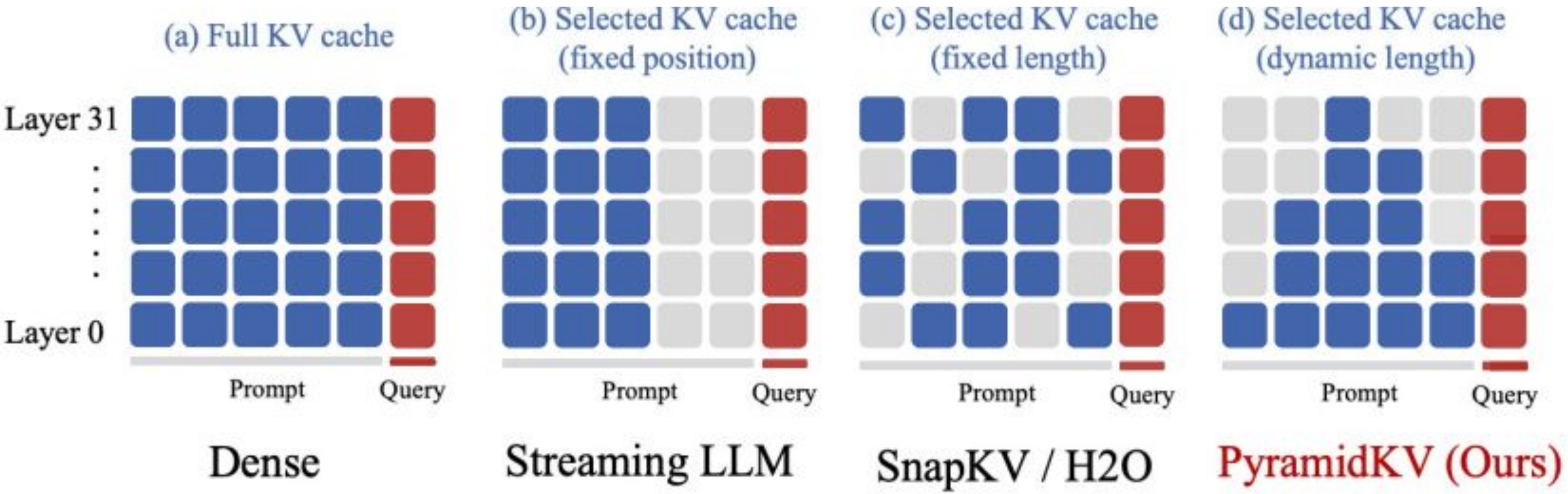
- Which tokens do we keep?
  - Instruction tokens - Keep
  - Compute how much each token is attended to by the instruction tokens
  - Keep the tokens with the highest attention scores

$$s_i^h = \sum_{j \in [n-\alpha, n]} A_{ij}^h$$

Experiments & Results

Setup

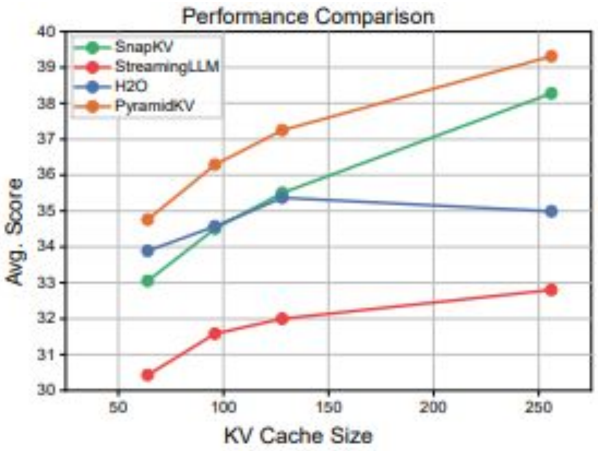
- Models: LLaMA-3-8B, LLaMA-3-70B, Mistral-7B
- Benchmark: LongBench (17 datasets across QA, summarization, code, few-shot learning).
- Baselines: FullKV, StreamingLLM, H2O, SnapKV.
- Same total KV budget on average across methods.



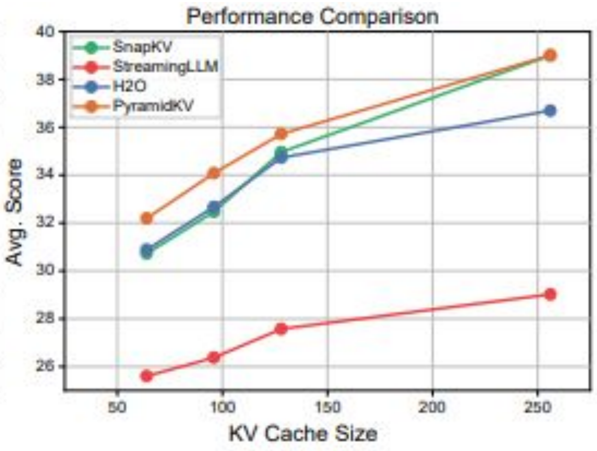
Results

- PyramidKV consistently outperforms baselines, especially with small cache sizes.
- Maintains near-full performance using only 12% of full KV cache
- Even with 0.7% of KV cache, accuracy drop is minimal

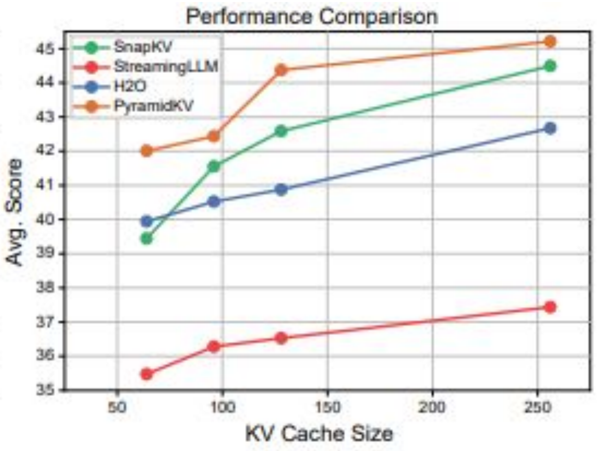
LLaMA-3-8B



Mistral-7B



LLaMA-3-70B



Avg score across datasets for 64, 96, 128, and 256 cache sizes

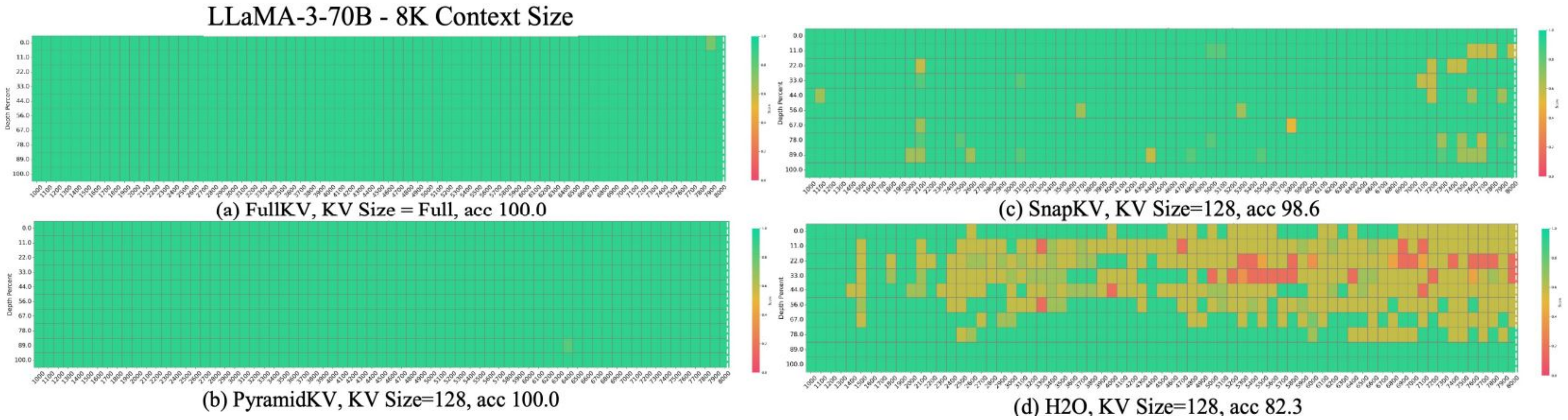
Results Cont'd

- PyramidKV excels with small cache sizes:

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	Lcc	RB-P	
	18409	3619	4559	9151	4887	11214	8734	10614	2113	5177	8209	6258	11141	9289	1235	4206	
LlaMa-3-70B-Instruct, KV Size = Full																	
FKV	27.75	46.48	49.45	52.04	54.90	30.42	32.37	22.27	27.58	73.50	92.46	45.73	12.50	72.50	40.96	63.91	46.55
LlaMa-3-70B-Instruct, KV Size = 64																	
SKV	23.92	31.09	36.54	46.66	50.40	25.30	18.05	21.11	19.79	41.50	**91.06**	40.26	12.00	**72.50**	43.33	57.62	39.45
SLM	22.07	23.53	27.31	43.21	**51.66**	23.85	16.62	19.74	15.20	39.50	76.89	33.06	12.00	**72.50**	40.23	50.20	35.47
H2O	25.45	34.64	33.23	**48.25**	50.30	24.88	20.03	21.50	21.39	42.00	90.36	**41.58**	12.00	71.50	43.83	**58.16**	39.94
Ours	**25.47**	**36.71**	**42.29**	47.08	46.21	**28.30**	**20.60**	**21.62**	**21.62**	**64.50**	89.61	41.28	**12.50**	**72.50**	**45.34**	56.50	**42.01**
LlaMa-3-70B-Instruct, KV Size = 2048																	
SKV	26.73	45.18	47.91	**52.00**	**55.24**	30.48	28.76	22.35	27.31	72.50	92.38	45.58	12.00	**72.50**	**41.52**	**69.27**	46.36
SLM	26.69	41.01	35.97	46.55	52.98	25.71	27.81	20.81	27.16	69.00	91.55	44.02	12.00	72.00	41.44	68.73	43.96
H2O	**27.67**	**46.51**	**49.54**	51.49	53.85	29.97	28.57	**22.79**	**27.53**	59.00	**92.63**	**45.94**	12.00	72.50	41.39	63.90	45.33
Ours	27.22	46.19	48.72	51.62	54.56	**31.11**	**29.76**	22.50	27.27	**73.50**	91.88	45.47	12.00	72.50	41.36	69.12	**46.55**

Needle In A Haystack Experiment

- Purpose: test long-context factual retrieval
- Result: LLaMa-3-70B achieves 100% accuracy with 128 KV entries using PyramidKV, matching full cache performance
- Significance: PyramidKV preserves long-range memory and retrieval ability.

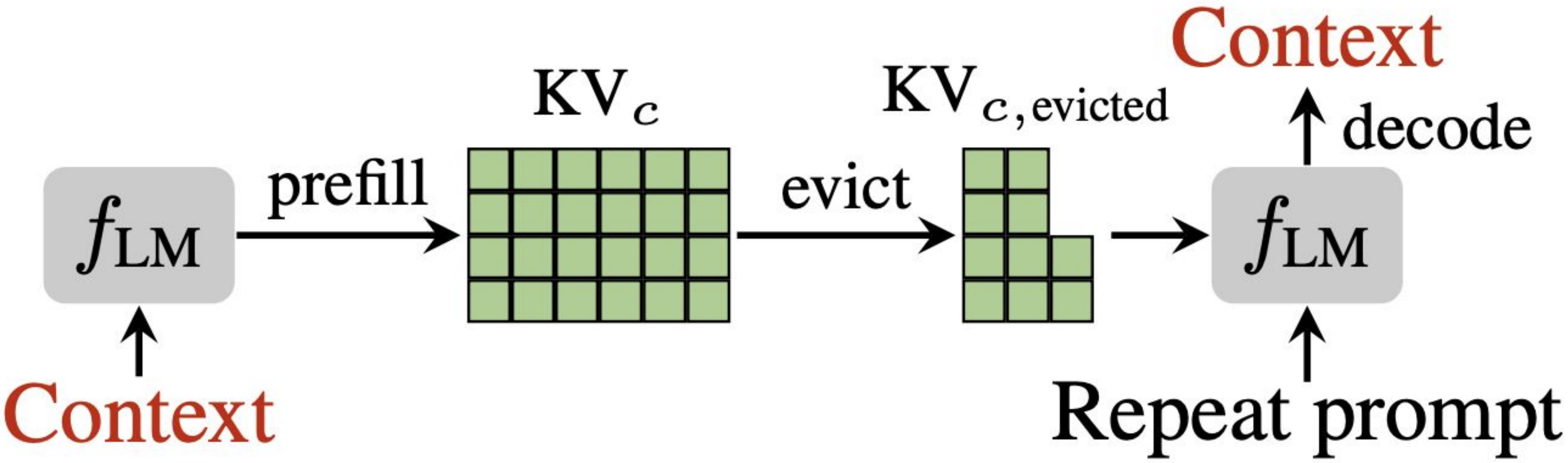


KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon,
Jae W. Lee , Sangdoo Yun, Hyun Oh Song

JOHNS HOPKINS
UNIVERSITY

KVzip

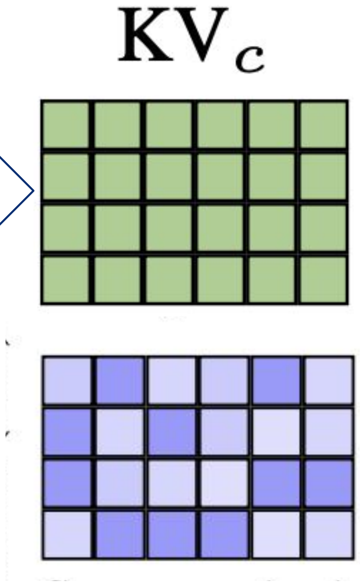


f_{LM}
Context
prefill
KV_c
evict
$\text{KV}_{c,\text{evicted}}$
f_{LM}
Repeat prompt
decode
Context

KVzip: How to evict?

Evict tokens that don't contribute to reconstructing the context

Here is some context to compress

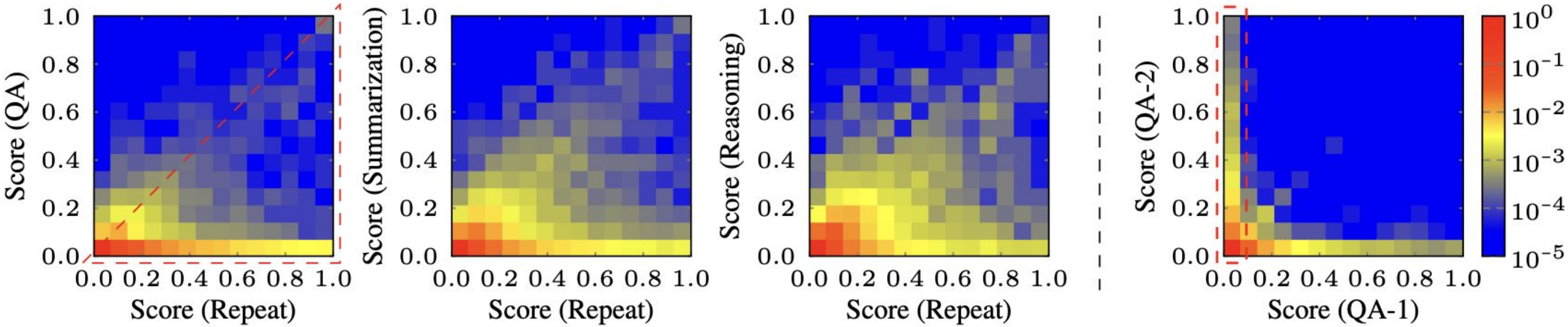


Here is some context to compress

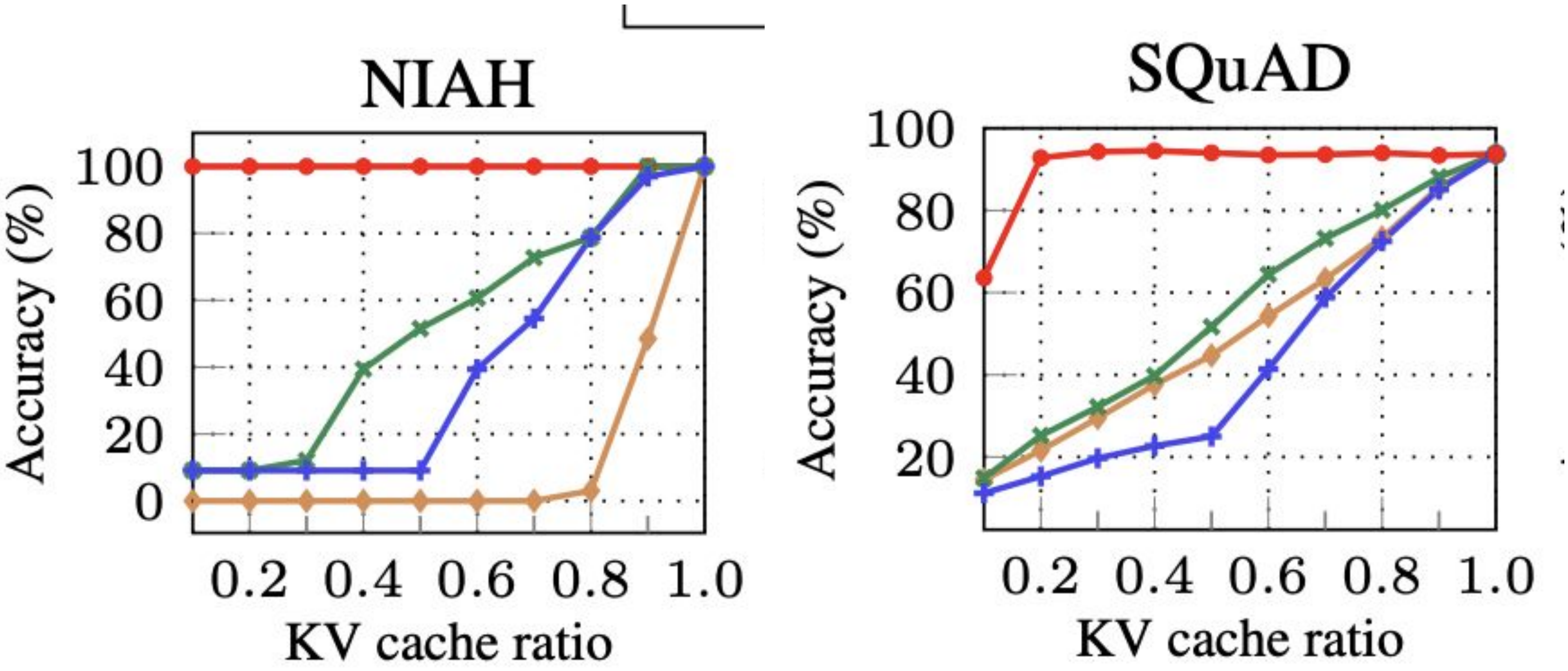
$$S_{l,h} = \max_{g=1,\ldots,G;\ i=1,\ldots,n_{\text{in}}} \bar{A}_{l,h}[g,i].$$

Importance

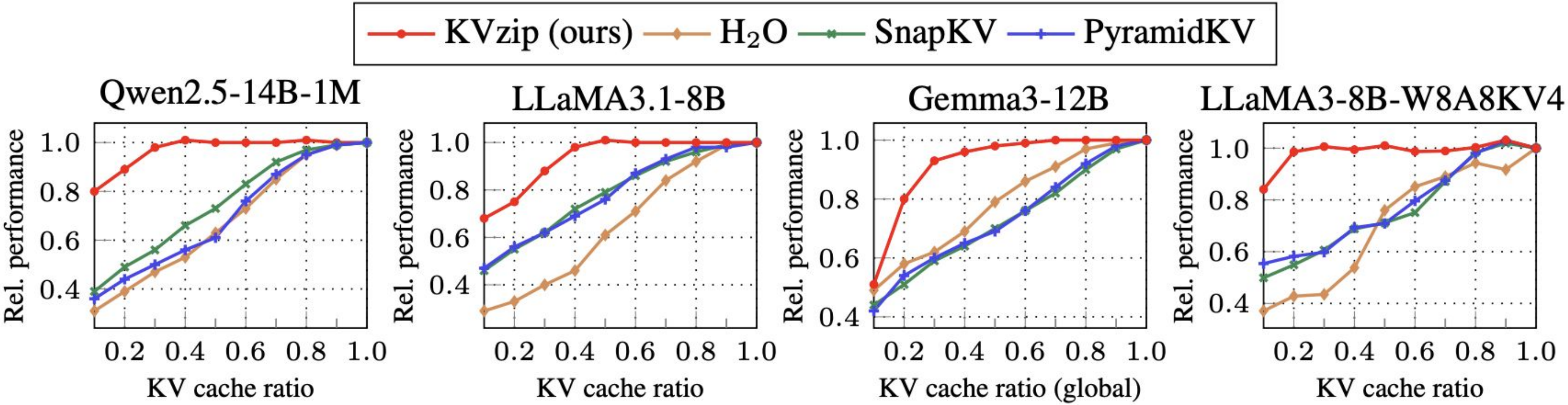
KVzip: Attention Sparsity



KVzip: Performance



KVzip: Model Comparison



Conclusions & Critiques

Conclusions

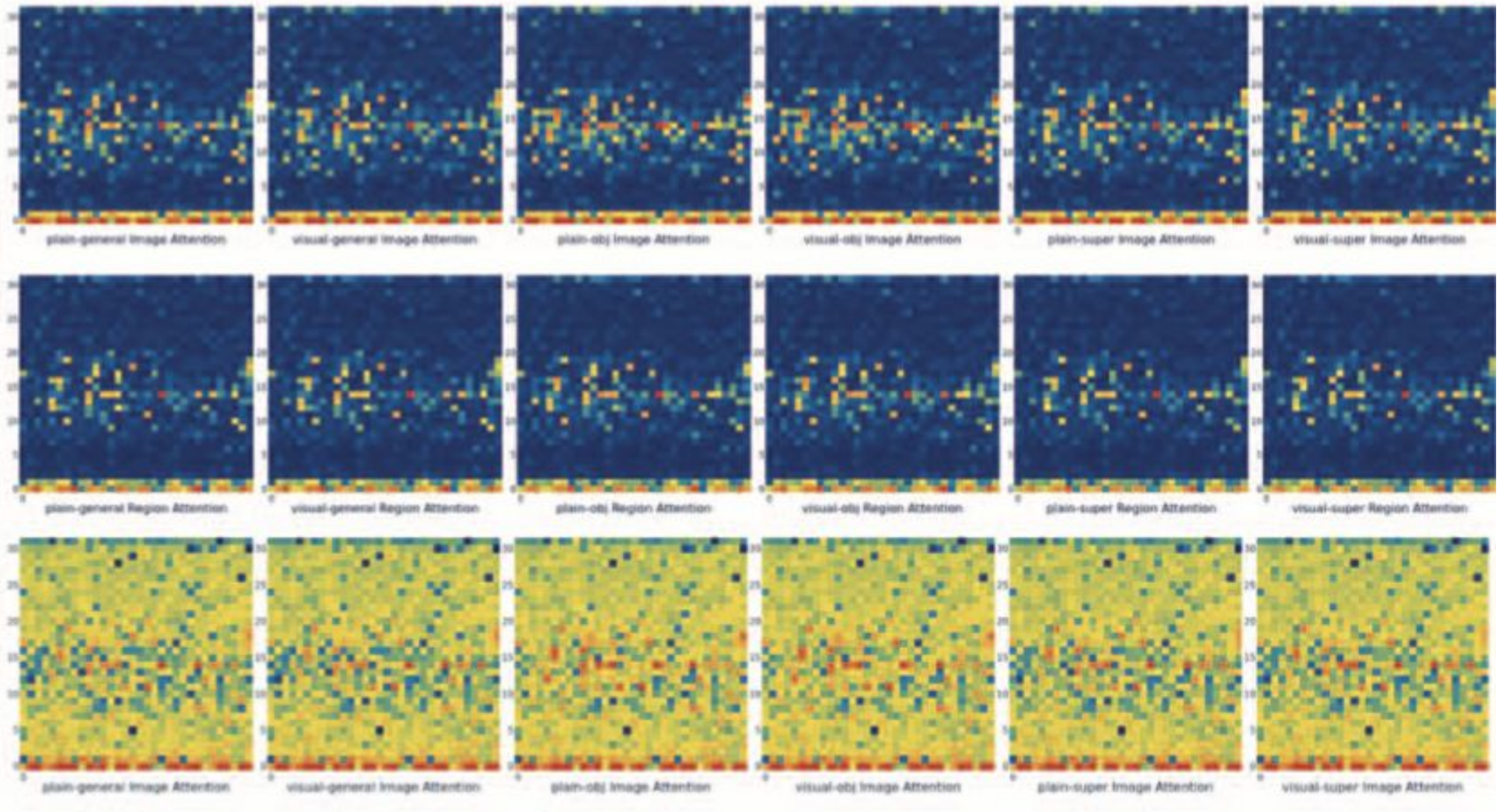
- Main takeaway: PyramidKV mirrors attention naturally funneling through layers
- Performance:
  - Preserves accuracy while using only 12% of KV cache
  - Preserves long-context understanding ability
- Efficiency: Up to 90% GPU-memory reduction, minimal runtime overhead:

cache size	Memory	Compression Ratio	QMSum	TREC	TriviaQA	PCount	PRe	Lcc
512	428M	6.3%	22.80	71.50	90.61	5.91	69.50	58.16
1024	856M	12.5%	22.55	71.50	90.61	5.91	69.50	58.16
2048	1712M	25.0%	22.55	72.00	90.56	5.58	69.25	56.79
Full	6848M	100.0%	23.30	73.00	90.56	5.22	69.25	58.76

Critiques

- My thoughts: Very impressive results, lots of memory reduction with a simple implementation.
- Limits:
  - Evaluated only on 3 English models.
    - No multilingual testing.
  - The pyramid could fail on tasks with different attention shapes.
    - The observed attention phenomenon was only for multi-doc QA tasks.

Critiques: Attention Sparsity, Do you really want that?



Thank You!

Any Questions?

Gabriel Pernell, Alexander Martin