SparseMM: Head Sparsity Emerges from Visual Concept Responses in MLLMs

Supplementary Material

1. Implementation details

1.1. Implementation details about GQA

The models LLaVA-NeXT-Mistral-7B, and Qwen2-VL-7B-Instruct are Grouped-Query Attention (GQA) models, which differ markedly from conventional multi-head attention (MHA) mechanisms in the computation of attention. In a GQA model, the query state is of shape $(bs, seq\_len, num\_query\_heads, hidden\_dim)$, while the key and value states, collectively forming the KV cache, are of shape $(bs, seq\_len,$ $num\_key\_value\_heads, hidden\_dim)$. During the attention calculation, the key and value states are repeated

$$\frac{num\_query\_heads}{num\_key\_value\_heads} = num\_key\_value\_group$$

times, thereby restoring the setup analogous to MHA. Prior to computation, the sequence length dimension and the query head dimension are interchanged, resulting in an attention score tensor of shape $(bs, num\_query\_heads, seq\_len, seq\_len)$. Subsequently, when this tensor is combined with the value states, the output is of shape $(bs, num\_query\_heads, seq\_len, hidden\_dim)$

From the above reasoning, it follows that we obtain a visual head score matrix with dimensions $(layers, num\_query\_head)$. This is the origin of the score distribution depicted in Fig. 2.

In practical scenarios involving the preservation of the Key-Value cache, each key-value head is associated with num_key_value_group attention scores. The total attention score for a given head is computed as the sum of the scores of the corresponding group. This aggregate score is then employed for the allocation of the budget for the Key-Value cache.

1.2. Details on Evaluation Metrics

We adopt different evaluation metrics for different benchmarks. For the DocVQA [6] benchmark, we employ the ANLS metric. This metric evaluates the similarity between the predicted answer and the ground truth by normalizing the Levenshtein distance, thereby accommodating minor variations in format and phrasing while maintaining a robust assessment of answer quality. For the OCRBench [4], TextVQA [8], MMBench [3], GQA [1] and VQAv2 [2] benchmark, we use accuracy as the primary metric. For ChartQA [5] benchmark, we utilize the relaxed accuracy metric. This measure provides partial credit for responses that are close to the ground truth, thereby offering a more nuanced perspective on model performance when outputs are not perfectly correct but still largely informative. Finally, for the TextCaps [7] dataset, we adopt the CIDEr metric. CIDEr assesses the quality of generated captions by computing a weighted n-gram similarity between the candidate and reference captions.

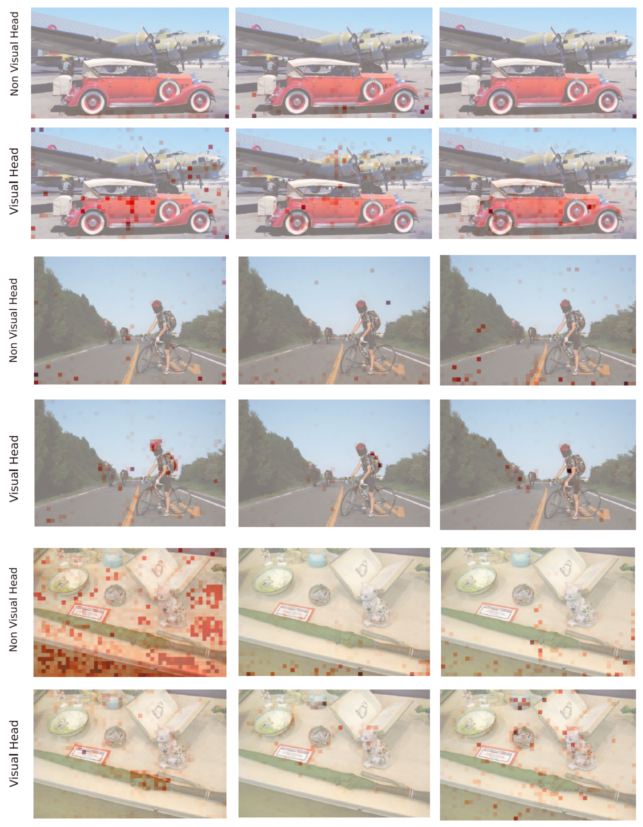

Figure 1. **More Visualization Results.** Visual heads are able to attend to the correct objects, whereas non-visual heads cannot.

2. More Visualization

We conduct more visualization on the visual head in Fig. 1. We use LLaVA-NeXT-Vicuna-7B model for the experiment.

3. More Analysis

Ablations on Budget Allocation Ratios. We conducted an ablation study on the hyperparameter ρ. This study evaluated the performance of three models on OCRBench, with a budget of 256. The results are presented in Tab. 1. For the LLaVA-NeXT-Vicuna-7B model, the ratio $\rho = 0.1$ achieved the highest performance score of 0.522, outperforming other ratios. Similarly, for the LLaVA-NeXT-Mistral-7B model,

Table 1. **Ablation on Budget Allocation Ratios.** We conducted an ablation study on the hyperparameter ρ and the results indicated that the performance is optimal when the ratio is set to 0.1. Therefore, we use 0.1 as the default value in our experiments.

Ratio ρ	0	0.1	0.2	0.3	0.4	0.5	0.8	1.0
LLaVA-NeXT-Vicuna-7B	0.507	**0.522**	0.520	0.520	0.516	0.515	0.510	0.460
LLaVA-NeXT-Mistral-7B	0.145	**0.519**	0.517	0.514	0.514	0.518	0.506	0.451
Qwen2-VL-7B-Instruct	0.809	**0.812**	0.811	0.808	0.807	0.804	0.789	0.775

Table 2. **Ablation on Cache Allocation Strategies.** The results demonstrate that each of the three cache components plays an essential role and that none can be omitted without negatively impacting overall performance.

Local Window	Uniform-Based	Score-Preferred	MMBench					
Cache	Cache	Cache	512	256	128	96	64	48
✓	✗	✗	81.3	80.5	77.3	73.6	70.5	67.2
✓	✓	✗	81.5	81.4	79.3	77.6	74.6	73.9
✓	✓	✓	81.5	81.4	81.5	81.4	80.3	77.9

a ratio of 0.1 also resulted in a peak performance score of 0.519, which is significantly higher compared to the scores at other ratios. While the Qwen2-VL-Instruct model exhibited only a marginally higher score at $\rho = 0.1$ (0.812), this still represents the highest performance across all tested ratios. It is noteworthy that the Mistral model exhibits a significant performance drop at a ratio of $\rho = 0$. This observation suggests that relying entirely on visual head score allocation of the cache budget can result in some heads being unable to attend to any preceding input information. Consequently, this underscores the necessity of assigning a Uniform-Based cache to each head. By ensuring that each head receives a guaranteed share of the cache resources, we can prevent such performance degradation and enhance the overall effectiveness of the model.

Ablation on Cache Allocation Strategies. We add an ablation study on Qwen2-VL-7B-Instruct to investigate the effectiveness of the three-part cache allocation mechanism. As shown in Tab. 2, using only Local-Window Cache limits its context and causes larger drops with smaller budgets. Combining Local-Window and Uniform-Based Caches lacks head-level allocation and underperforms compared to our SparseMM.

4. Numerical results

We present the numerical results of our main experimental results for reference and further research.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023. 1

[2] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017. 1

[3] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023. 1

[4] Yuliang Liu, Zhang Li, Biao Yang, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. On the hidden mystery of ocr in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023. 1

[5] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 1

[6] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021. 1

[7] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 742–758. Springer, 2020. 1

[8] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 1

Table 3. Numerical results of Fig. 4.

Benchmark	Method	LLaVA-NeXT-Vicuna-7B						LLaVA-NeXT-Mistral-7B						Qwen2-VL-7B-Instruct					
		2048	1024	512	256	128	64	2048	1024	512	256	128	64	2048	1024	512	256	128	64
DocVQA	SparseMM	0.6841	0.6837	0.6811	0.6784	0.6677	0.6377	0.6310	0.6272	0.6227	0.6163	0.6082	0.5756	0.9394	0.9392	0.9394	0.9345	0.9154	0.8493
	SnapKV	0.6845	0.6807	0.6709	0.6430	0.5906	0.4977	0.6365	0.6306	0.6215	0.5971	0.5519	0.4726	0.9384	0.9340	0.9194	0.8798	0.8012	0.6652
	PyramidKV	0.6843	0.6812	0.6714	0.6494	0.6030	0.4901	0.6363	0.6225	0.6076	0.5818	0.5425	0.4351	0.9391	0.9343	0.8816	0.8180	0.7394	0.5990
	AdaKV	0.6839	0.6823	0.6753	0.6526	0.6064	0.5411	0.6358	0.6299	0.6174	0.5957	0.5592	0.4872	0.9392	0.9330	0.9201	0.8841	0.8121	0.6847
	Random	0.6834	0.6791	0.6646	0.6340	0.5816	0.4868	0.6326	0.6217	0.5971	0.5592	0.4973	0.4206	0.9275	0.9015	0.8534	0.7681	0.6963	0.5102
OCRBench	SparseMM	0.519	0.522	0.528	0.523	0.501	0.478	0.523	0.518	0.512	0.519	0.507	0.462	0.821	0.822	0.821	0.812	0.795	0.743
	SnapKV	0.525	0.518	0.510	0.461	0.412	0.340	0.529	0.517	0.500	0.450	0.390	0.319	0.819	0.813	0.801	0.773	0.719	0.624
	PyramidKV	0.525	0.524	0.502	0.476	0.409	0.312	0.528	0.512	0.489	0.440	0.394	0.290	0.820	0.814	0.776	0.739	0.682	0.563
	AdaKV	0.524	0.517	0.508	0.484	0.430	0.351	0.529	0.520	0.502	0.451	0.404	0.328	0.819	0.812	0.796	0.778	0.710	0.621
	Random	0.523	0.516	0.500	0.458	0.397	0.320	0.521	0.507	0.451	0.399	0.354	0.261	0.811	0.794	0.760	0.704	0.633	0.489
TextVQA	SparseMM	0.6499	0.6492	0.6474	0.6470	0.6417	0.6312	0.6555	0.6547	0.6531	0.6505	0.6474	0.6281	0.8213	0.8215	0.8203	0.8218	0.8164	0.7719
	SnapKV	0.6488	0.6474	0.6408	0.6229	0.6010	0.5616	0.6565	0.6541	0.6503	0.6345	0.6103	0.5712	0.8213	0.8212	0.8204	0.8031	0.7746	0.6990
	PyramidKV	0.6487	0.6483	0.6410	0.6277	0.6040	0.5502	0.6566	0.6490	0.6430	0.6285	0.6088	0.5467	0.8218	0.8218	0.8076	0.7774	0.7440	0.6547
	AdaKV	0.6482	0.6486	0.6429	0.6199	0.5988	0.5609	0.6566	0.6530	0.6464	0.6289	0.6049	0.5685	0.8213	0.8212	0.8185	0.7985	0.7695	0.7025
	Random	0.6478	0.6438	0.6373	0.6235	0.6011	0.5653	0.6536	0.6494	0.6358	0.6134	0.5822	0.5400	0.8202	0.8166	0.7943	0.7601	0.6955	0.5852
ChartQA	SparseMM	0.5480	0.5452	0.5488	0.5392	0.5380	0.5276	0.5280	0.5216	0.5236	0.5188	0.5116	0.4888	0.8152	0.8152	0.8128	0.8160	0.8152	0.8016
	SnapKV	0.5480	0.5536	0.5416	0.5000	0.4527	0.4304	0.5288	0.5236	0.5164	0.5016	0.4752	0.4272	0.8140	0.8144	0.8144	0.8128	0.7964	0.7552
	PyramidKV	0.5488	0.5536	0.5496	0.5304	0.4716	0.4100	0.5272	0.5228	0.5080	0.4920	0.4708	0.4068	0.8140	0.8144	0.8144	0.8088	0.7924	0.7332
	AdaKV	0.5492	0.5540	0.5480	0.4912	0.4576	0.4384	0.5292	0.5224	0.5156	0.5044	0.4780	0.4460	0.8152	0.8156	0.8140	0.8080	0.7964	0.7592
	Random	0.5480	0.5476	0.5424	0.5304	0.4936	0.4372	0.5272	0.5152	0.5060	0.4764	0.4428	0.3944	0.8152	0.8152	0.8060	0.7876	0.7500	0.6696
TextCaps	SparseMM	0.7320	0.7309	0.7334	0.7284	0.7071	0.5992	0.7067	0.7054	0.6896	0.6795	0.6339	0.5238	1.4697	1.4744	1.4919	1.4915	1.4299	1.0431
	SnapKV	0.7226	0.7167	0.6969	0.6495	0.5642	0.4431	0.7070	0.6969	0.6970	0.6504	0.5579	0.4436	1.4677	1.4744	1.4695	1.3598	1.1424	0.7940
	PyramidKV	0.7237	0.7254	0.6953	0.6491	0.5745	0.4164	0.7061	0.6828	0.6592	0.6230	0.5495	0.4062	1.4694	1.4680	1.2745	1.1151	0.9536	0.5669
	AdaKV	0.7263	0.7273	0.7039	0.6598	0.5923	0.4727	0.7037	0.6953	0.6850	0.6459	0.5664	0.4400	1.4690	1.4650	1.4631	1.3445	1.1461	0.8133
	Random	0.7297	0.7219	0.6803	0.6268	0.5355	0.4356	0.7065	0.6980	0.6882	0.6472	0.5512	0.4368	1.4690	1.4727	1.4812	1.3824	1.1627	0.8116

Table 4. Numerical results of Fig. 5.

Method	MMBench						GQA						VQAv2					
	512	256	128	96	64	48	512	256	128	96	64	48	512	256	128	96	64	48
SparseMM	81.52	81.44	81.52	81.44	80.33	77.92	64.51	64.52	64.20	63.66	62.48	60.88	75.46	75.46	75.24	75.06	74.58	74.36
SnapKV	81.52	81.44	79.64	77.75	74.57	73.79	64.53	64.51	63.77	62.38	60.82	59.19	75.38	75.50	75.02	74.32	73.58	71.98
PyramidKV	81.53	79.64	76.46	74.14	73.45	73.30	63.80	63.47	62.05	60.65	59.41	59.37	75.38	75.30	74.72	73.60	71.60	68.88
AdaKV	81.52	81.44	79.81	77.83	75.17	73.45	64.52	64.65	63.52	62.55	61.59	59.20	75.40	75.34	75.14	74.08	73.66	72.02
Random	81.52	81.36	79.64	77.92	74.22	73.54	64.51	64.38	63.87	62.60	61.00	59.39	75.28	75.32	74.78	74.16	73.36	72.44