Extending Context Window of Large Language Models via Positional Interpolation

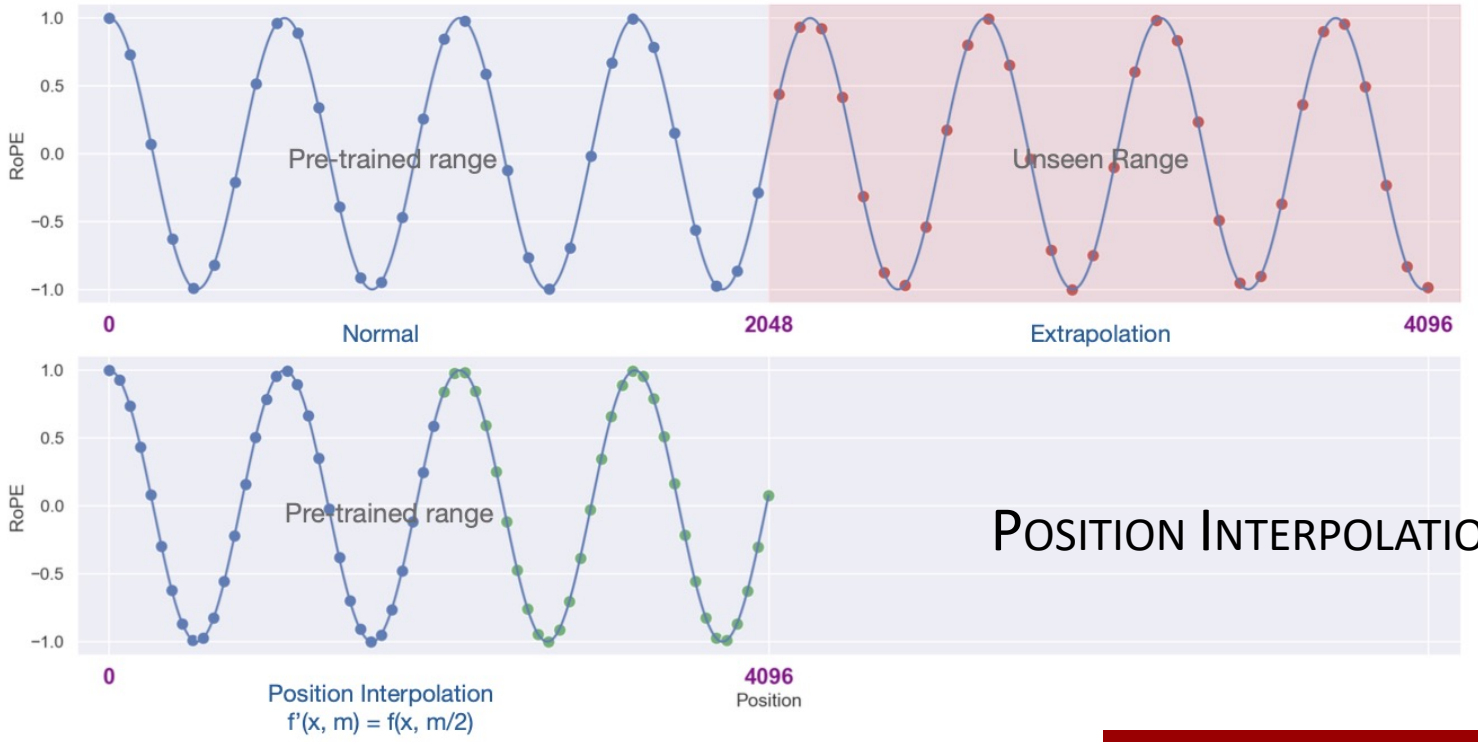
Shouyuan Chen **Sherman Wong** **Liangjian Chen** **Yuandong Tian**
Meta Platforms Inc.
{chenshouyuan,shermanwong,clj,yuandong}@meta.com

author={Chen, Shouyuan and Wong, Sherman and Chen, Liangjian and Tian, Yuandong},
journal={arXiv preprint arXiv:2306.15595},
year={2023.06.28}
Cites={113}

Method

Size	Model Context Window	Method	Fine-tuning steps 200	400	600	800	1000	10000
7B	8192	FT	1792	2048	2048	2048	2304	2560
33B	8192	FT	1792	2048	1792	2048	2304	-
7B	8192	PI	8192	8192	8192	8192	8192	-
7B	16384	PI	16384	16384	16384	16384	16384	-
7B	32768	PI	32768	32768	18432	32768	32768	-
33B	8192	PI	8192	8192	8192	8192	8192	-
33B	16384	PI	16384	16384	16384	16384	16384	-

Table 4: Effective context window sizes after fine-tuning. FT: Direct fine-tuning. PI: Position Interpolation.



POSITION INTERPOLATION

位置编码只依赖于相对位置

Transformer models require explicit positional information to be injected, typically in the form of positional encodings, to represent the order of inputs. We consider Rotary Position Embedding (RoPE) (Su et al., 2021), which is the position encoding used in the LLaMA model (Touvron et al., 2023). Given a position index $m \in [0, c)$ and an embedding vector $\mathbf{x} := [x_0, x_1, \ldots, x_{d-1}]^\top$, where d is the dimension of the attention head, RoPE defines a vector-valued complex function $\mathbf{f}(\mathbf{x}, m)$ as follows

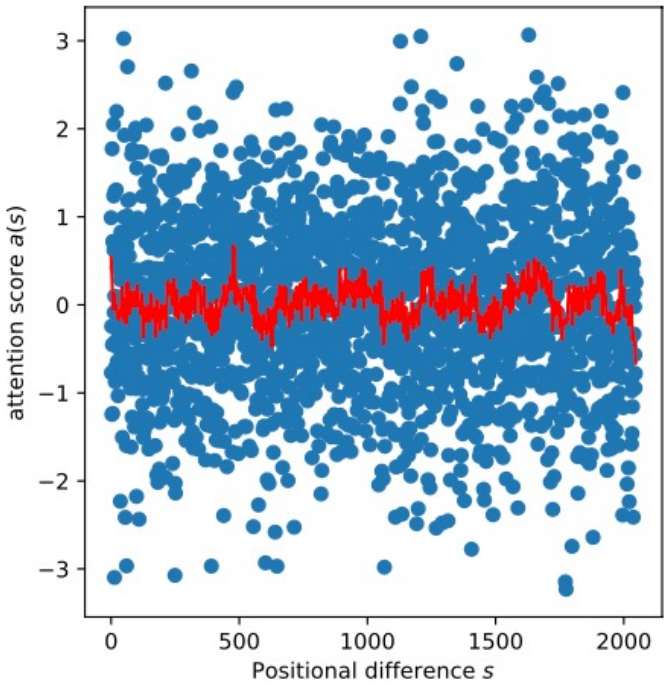
$$\mathbf{f}(\mathbf{x}, m) = [(x_0 + \mathrm{i}x_1)e^{\mathrm{i}m\theta_0}, (x_2 + \mathrm{i}x_3)e^{\mathrm{i}m\theta_1}, \ldots, (x_{d-2} + \mathrm{i}x_{d-1})e^{\mathrm{i}m\theta_{d/2-1}}]^\top \tag{1}$$

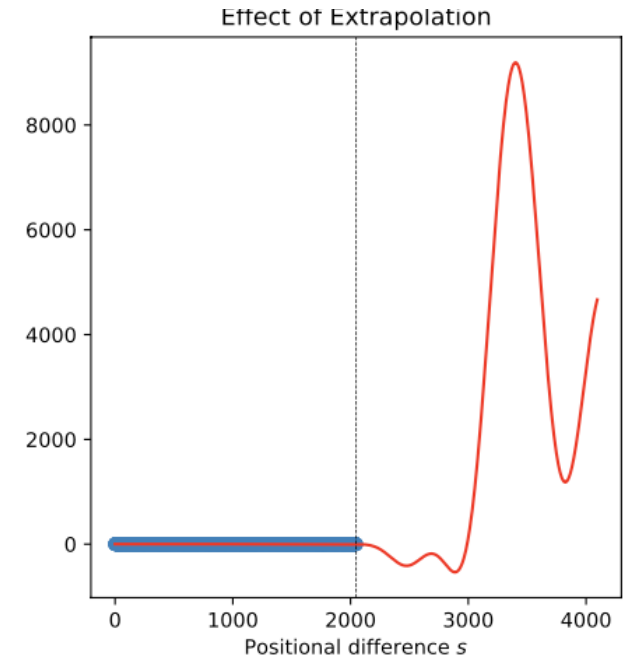
where $\mathrm{i} := \sqrt{-1}$ is the imaginary unit and $\theta_j = 10000^{-2j/d}$. Using RoPE, the self-attention score

$$\begin{aligned} a(m, n) &= \mathrm{Re}\langle \mathbf{f}(\mathbf{q}, m), \mathbf{f}(\mathbf{k}, n)\rangle \\ &= \mathrm{Re}\left[\sum_{j=0}^{d/2-1} (q_{2j} + \mathrm{i}q_{2j+1})(k_{2j} - \mathrm{i}k_{2j+1})e^{\mathrm{i}(m-n)\theta_j}\right] \\ &= \sum_{j=0}^{d/2-1} (q_{2j}k_{2j} + q_{2j+1}k_{2j+1})\cos((m-n)\theta_j) + (q_{2j}k_{2j+1} - q_{2j+1}k_{2j})\sin((m-n)\theta_j) \\ &=: a(m-n) \end{aligned} \tag{2}$$

is only dependent on relative position $m - n$ through trigonometric functions. Here $\mathbf{q}$ and $\mathbf{k}$ are the query and key vector for a specific attention head. At each layer, RoPE is applied on both query and key embeddings for computing attention scores.

$$a(s) = \mathrm{Re}\left[\sum_{j=0}^{d/2-1} h_j e^{\mathrm{i}s\theta_j}\right] \qquad h_j := (q_{2j} + \mathrm{i}q_{2j+1})(k_{2j} - \mathrm{i}k_{2j+1})$$





Method

证明Positional Interpolation的上界比Extrapolation 小很多

In Fig. 2, thanks to the smoothness of bases functions ϕ_j *interpolation* is much more stable and will not lead to wild values. Therefore, instead of extrapolate the attention score in Eqn. 3 to $s > L$, how about we define an attention score $\tilde{a}(s) = a(Ls/L')$ where L' is the longer context window? Formally, we replace RoPE $\mathbf{f}$ by $\mathbf{f}'$ defined as follows

$$\mathbf{f}'(\mathbf{x}, m) = \mathbf{f}\left(\mathbf{x}, \frac{mL}{L'}\right). \tag{4}$$

Positional Interpolation的上界

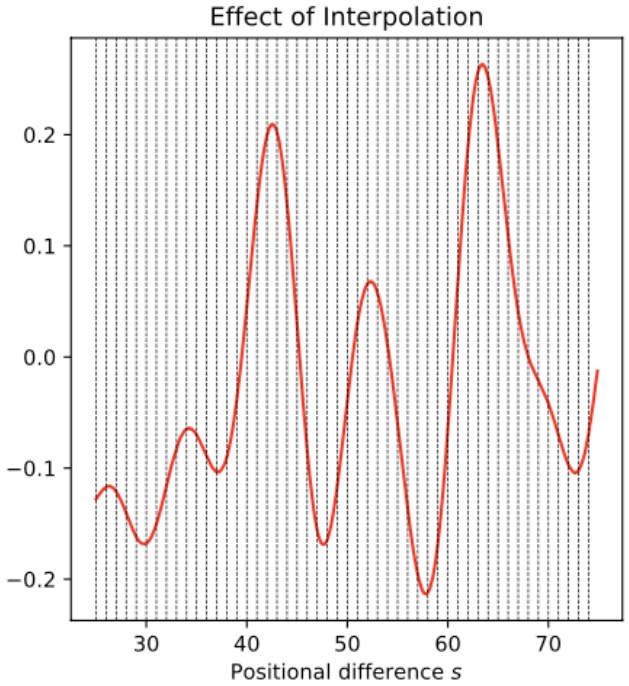
$$a(s) = \mathrm{Re}\left[\sum_{j=0}^{d/2-1} h_j e^{\mathrm{i}s\theta_j}\right] \qquad s \in [s_1, s_2]$$

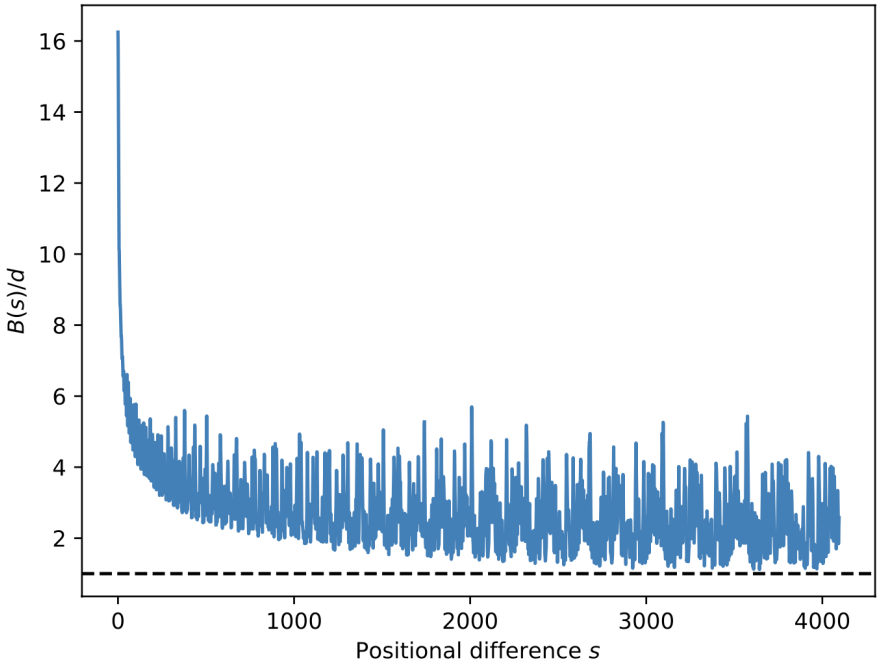
$$|a(s) - a_{\mathrm{linear}}(s)| \leq d\left(\max_j |h_j|\right)\frac{(s-s_1)(s_2-s)}{8\ln c}$$

$$|a(s) - a_{\mathrm{linear}}(s)| \leq \frac{d}{32\ln c}\max_j |h_j| \approx \frac{d\max_j |h_j|}{294.73}$$

Extrapolation的上界

$$|a(s)| \leq \left(\max_j |h_j - h_{j+1}|\right)\sum_{k=0}^{d/2-1}|A_{k+1}(s)| \leq 2\left(\max_j |h_j|\right)\sum_{k=0}^{d/2-1}|A_{k+1}(s)|,$$





Method

Size	Model Context Window	Method	Evaluation Context Window Size 2048	4096	8192	16384	32768
7B	2048	None	7.20	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
7B	8192	FT	7.21	7.34	7.69	-	-
7B	8192	PI	7.13	6.96	6.95	-	-
7B	16384	PI	7.11	6.93	6.82	6.83	-
7B	32768	PI	7.23	7.04	6.91	6.80	6.77
13B	2048	None	6.59	-	-	-	-
13B	8192	FT	6.56	6.57	6.69	-	-
13B	8192	PI	6.55	6.42	6.42	-	-
13B	16384	PI	6.56	6.42	6.31	6.32	-
13B	32768	PI	6.54	6.40	6.28	6.18	6.09
33B	2048	None	5.82	-	-	-	-
33B	8192	FT	5.88	5.99	6.21	-	-
33B	8192	PI	5.82	5.69	5.71	-	-
33B	16384	PI	5.87	5.74	5.67	5.68	-
65B	2048	None	5.49	-	-	-	-
65B	8192	PI	5.42	5.32	5.37	-	-

Size	Model Context Window	Method	Evaluation Context Window Size 2048	4096	8192	16384	32768
7B	2048	None	2.77	-	-	-	-
7B	8192	FT	2.85	2.74	2.73	-	-
7B	8192	PI	2.79	2.57	2.39	-	-
7B	16384	PI	2.79	2.57	2.37	2.25	-
7B	32768	PI	2.82	2.59	2.39	2.24	2.48
13B	2048	None	2.66	-	-	-	-
13B	8192	FT	2.71	2.56	2.50	-	-
13B	8192	PI	2.67	2.47	2.30	-	-
13B	16384	PI	2.68	2.47	2.29	2.18	-
13B	32768	PI	2.68	2.46	2.28	2.15	2.35
33B	2048	None	2.49	-	-	-	-
33B	8192	FT	2.56	2.48	2.47	-	-
33B	8192	PI	2.50	2.32	2.18	-	-
33B	16384	PI	2.53	2.34	2.18	2.07	-
65B	2048	None	2.42	-	-	-	-
65B	8192	PI	2.43	2.26	2.12	-	-

Longlora: Efficient fine-tuning of long-context large language models

Yukang Chen [1] **Shengju Qian** [1] **Haotian Tang** [2] **Xin Lai** [1]
Zhijian Liu [2] **Song Han** [2,3] **Jiaya Jia** [1]
[1]CUHK [2]MIT [3]NVIDIA

@article{
Conference={ICLR 2024}
Cites={57}
Oral
8 8 6 6

Method

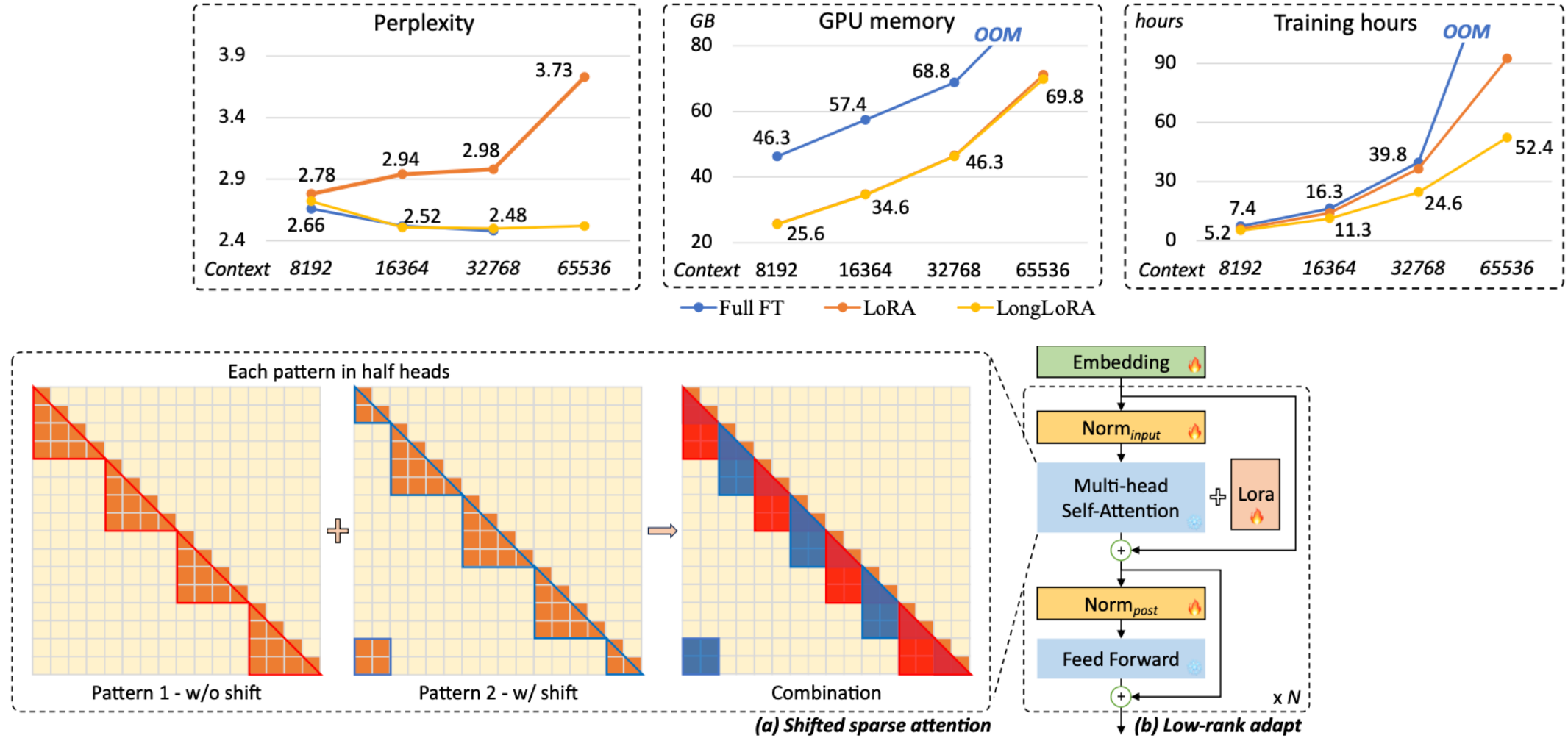
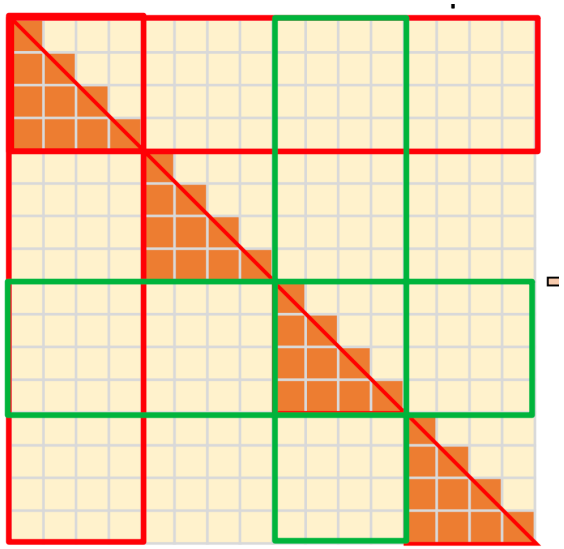
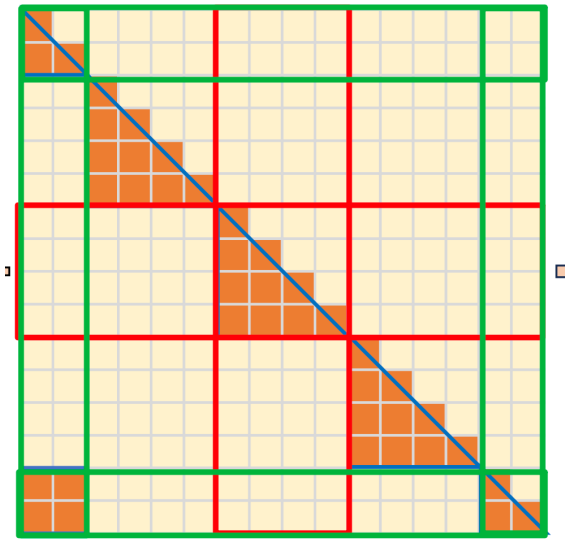


Figure 2: **Overview of LongLoRA**. We introduce Shifted Sparse Attention (S^2-Attn) during fine-tuning. The trained model retains original standard self-attention at inference time. In addition to training LoRA weights in linear layers, LongLoRA further makes embedding and normalization layers trainable. This extension is pivotal for context extension, and only introduces a minimal number of additional trainable parameters.

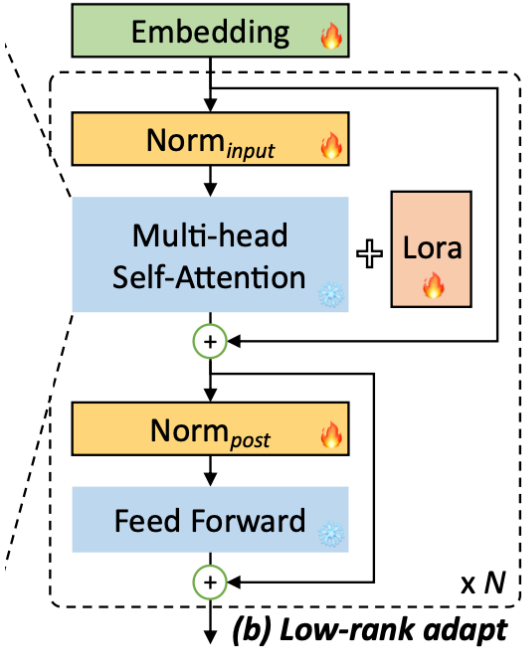
Method



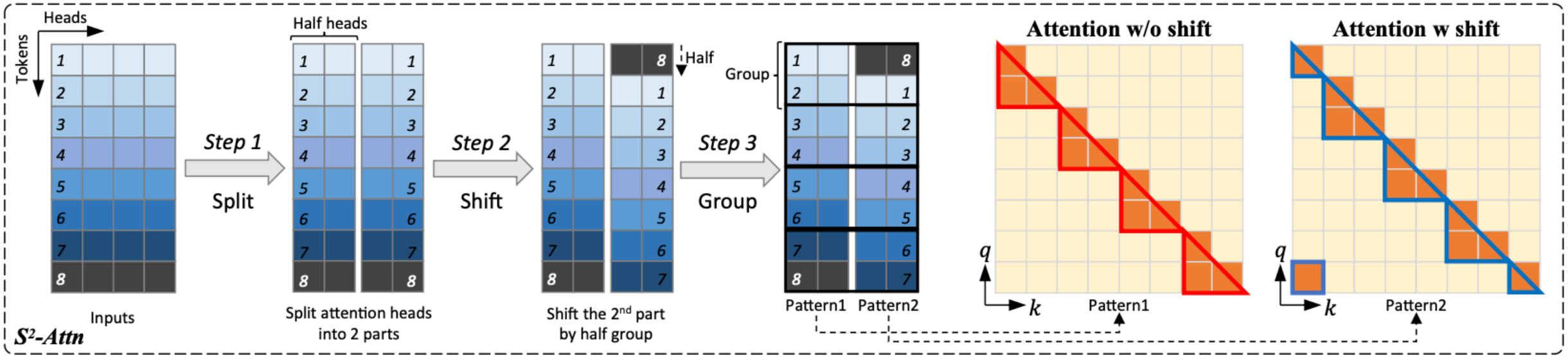
Pattern 1 - w/o shift



Pattern 2 - w/ shift

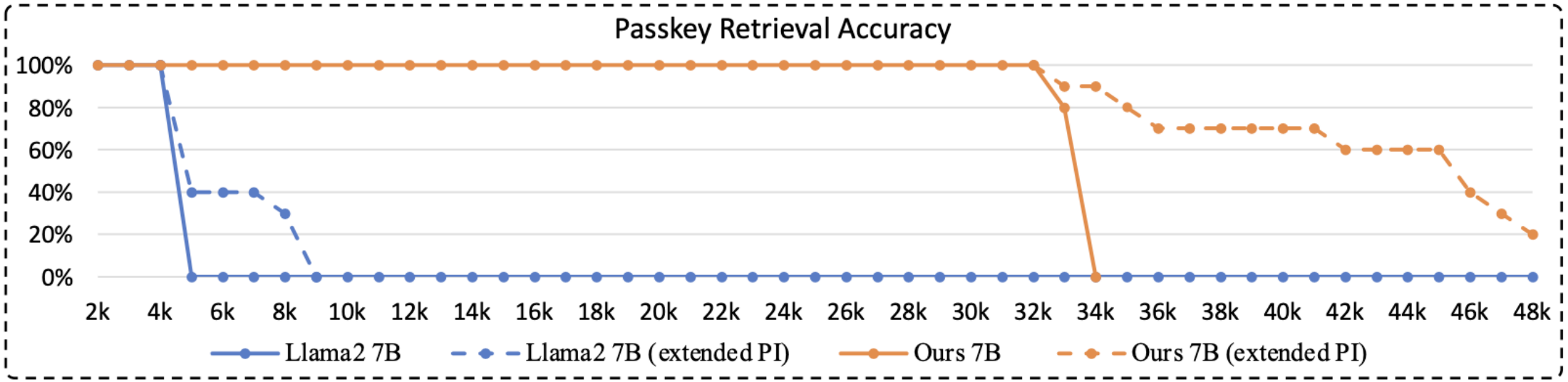


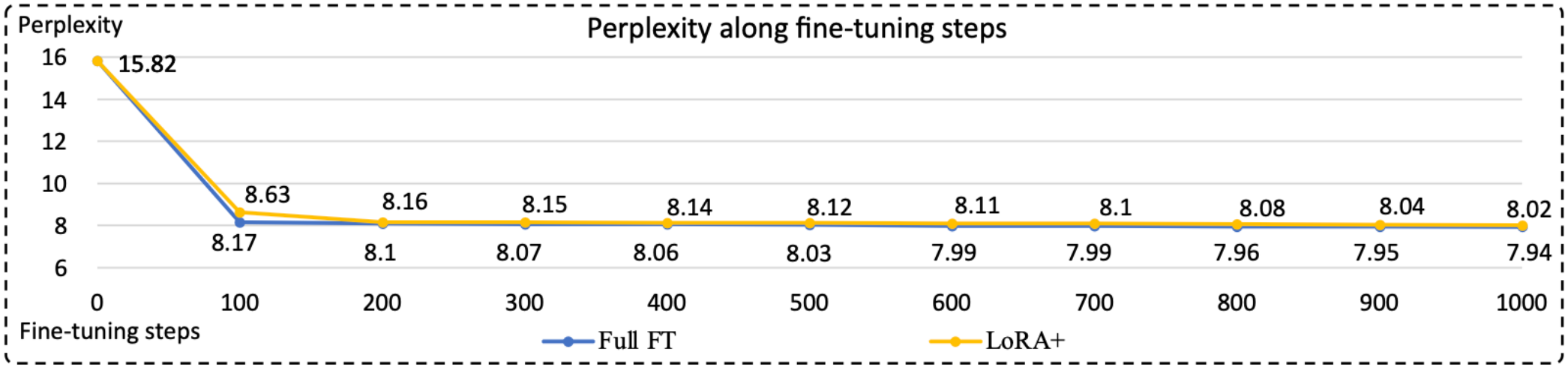
(b) Low-rank adapt



FXPFRIMFNT

Method	Full FT	LoRA (rank)						LoRA (rank = 8)		
		8	16	32	64	128	256	+ Norm	+ Embed	+ Norm & Embed
PPL	8.08	11.44	11.82	11.92	11.96	11.97	11.98	10.49	8.29	8.12





EXPERIMENT

在 8 * A100上LoRA

Size	Training Context Length	LongLoRA S^2-Attn	LongLoRA $LoRA^+$	Evaluation Context Length 2048	4096	8192	16384	32768
7B	8192			3.14	2.85	2.66	-	-
		✓		3.15	2.86	2.68	-	-
		✓	✓	3.20	2.91	2.72	-	-
	16384	✓		3.17	2.87	2.68	2.55	-
		✓	✓	3.17	2.87	2.66	2.51	-
	32768	✓		3.20	2.90	2.69	2.54	2.49
		✓	✓	3.35	3.01	2.78	2.61	2.50
13B	8192			2.96	2.69	2.53	-	-
		✓		3.01	2.74	2.57	-	-
		✓	✓	3.04	2.77	2.60	-	-
	16384	✓		2.99	2.72	2.53	2.40	-
		✓	✓	3.03	2.74	2.55	2.41	-
	32768	✓		3.04	2.75	2.56	2.42	2.33
		✓	✓	3.05	2.76	2.57	2.42	2.32

Evaluation Context	3k	6k	10k	13k	16k
ChatGLM2-6B (Du et al., 2022)	0.88	0.46	0.02	0.02	0.02
MPT-30B-chat (Team, 2023a)	0.96	**1.0**	0.76	-	-
MPT-7B-storywriter (Team, 2023b)	0.46	0.46	0.28	0.34	0.36
LongChat-13B (Li et al., 2023)	**1.0**	**1.0**	**1.0**	**0.98**	0.9
Ours-13B	**1.0**	0.98	0.98	**0.98**	**0.94**

Size	Training Context Length	Evaluation Context Length 2048	4096	8192	16384	32768	65536	100,000
7B	100,000	3.36	3.01	2.78	2.60	2.58	2.57	2.52
13B	65536	3.20	2.88	2.66	2.50	2.39	2.38	-
70B	32768	2.84	2.57	2.39	2.26	2.17	-	-

Model tells you what to discard: Adaptive kv cache compression for llms

Suyu Ge[1]*, **Yunan Zhang**[1]*, **Liyuan Liu**[2]*, **Minjia Zhang**[2], **Jiawei Han**[1], **Jianfeng Gao**[2]
[1]University of Illinois Urbana-Champaign, [2]Microsoft
{suyuge2,yunanz2,hanj}@illinois.edu
{lucliu,minjiaz,jfgao}@microsoft.com

@article{
Conference={ICLR 2024}
Cites={6}
Oral
8 8 8 8

Method

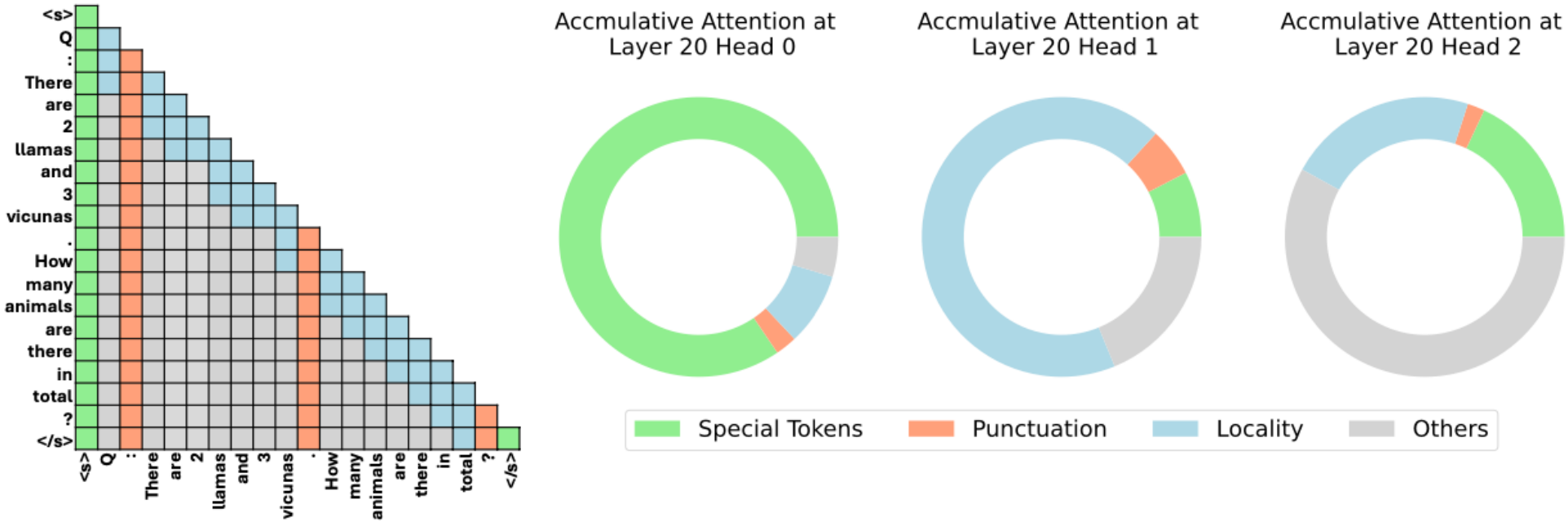


Figure 1: Different attention heads usually have different structures. Left: Four common attention structures (more details are elaborated in Section 3 and Section 4). Right: Attention map compositions of three attention heads that are in the same layer.

et al., 2023; Zhang et al., 2023; Liu et al., 2023a). We monitor for each token its cumulative sum of attention score, then treat these scores as token *frequency* and only keep the most frequent tokens in the KV cache. The length budget of frequent tokens over the current sequence length is controlled by a ratio r_f. This policy is referred to C_{frequent}.

Method

对于初始层和最终层，它们有更多分配给全KV缓存的注意力头，表明这些层的注意力头可能关注所有令牌。同时，对于中间层，注意力图集中在特殊令牌上，这表明这些层的大多数注意力头主要关注特殊令牌

Figure 3: Attention profiling result distribution across different layers.

Method

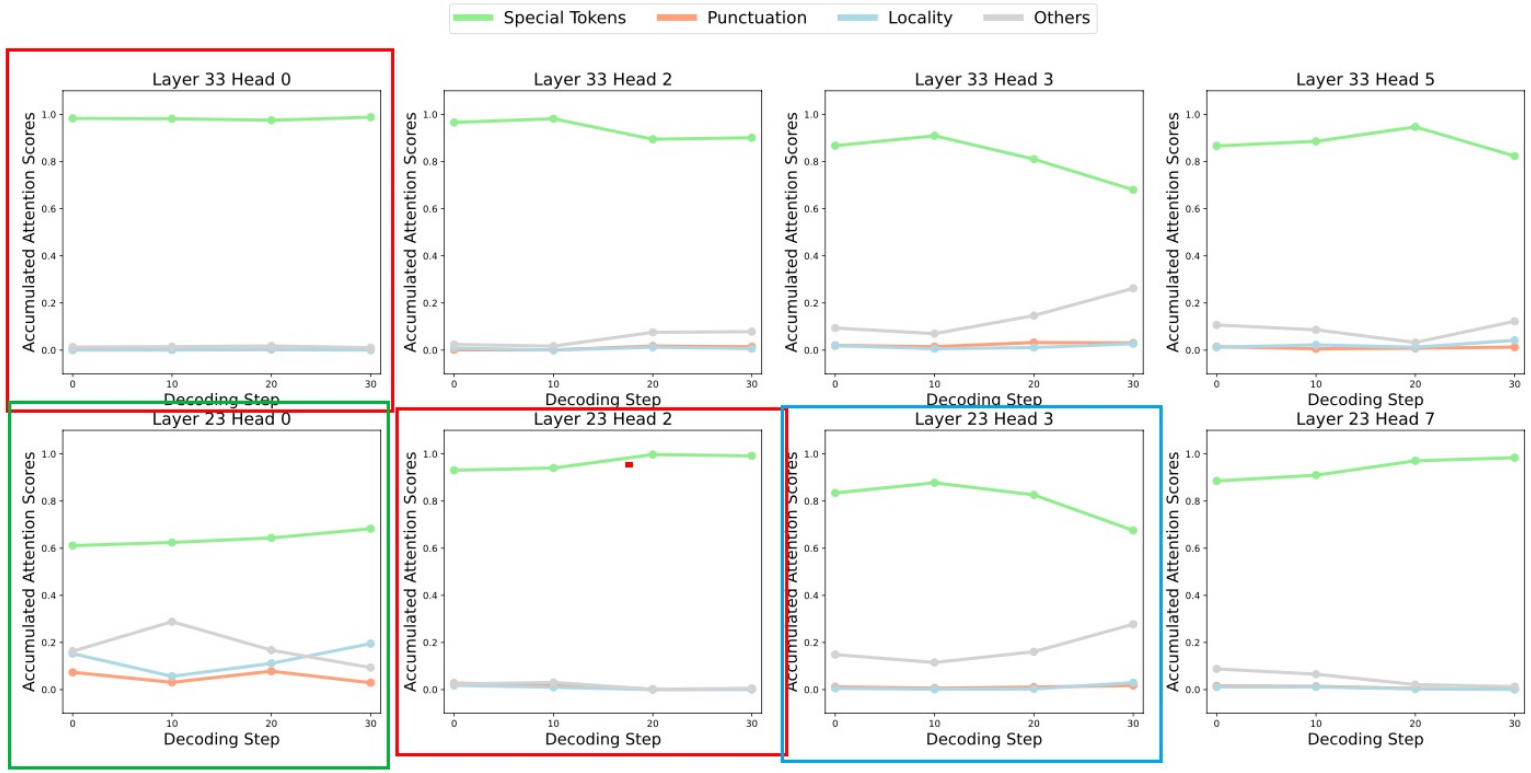


Figure 4: Accumulated attention score at 1st (prompt encoding), 10th, 20th, 30th decoding steps.

Observation. Despite some fluctuations of accumulated attention scores across time steps, the pattern of the attention maps remains relatively stable. For example, Layer 33 Head 0 and Layer 23 Head 2 almost only attend to the special token, while the locality and punctuation plays an important role in Layer 23 Head 0. As to Layer 23 Head 3, more than 10% of the attention score is allocated to the others portion, making it suitable for a uncompressed KV cache C_{full}.

Method

Algorithm 1: FastGen–Prompt Encoding.

Input: Feasible Policy Set ($\mathcal{C}$), Prompt
Output: Adaptive KV Cache

for *Attention Head* H_i *in LLM* **do**
  $\boldsymbol{K}^i, \boldsymbol{Q}^i, \boldsymbol{V}^i \leftarrow H_i(\text{Prompt})$
  $\boldsymbol{A}^i \leftarrow \text{softmax}(\boldsymbol{Q}^i {\boldsymbol{K}^i}^T)$
  $\boldsymbol{C}^i \leftarrow$ apply Equation 1 to $\boldsymbol{A}^i$
  /* $\mathtt{C}^i$: optimal policy */
  $\boldsymbol{K}^i_{\boldsymbol{C}^i}, \boldsymbol{V}^i_{\boldsymbol{C}^i} \leftarrow f(\boldsymbol{K}^i, \boldsymbol{V}^i, \boldsymbol{C}^i)$
  $\hat{\boldsymbol{K}}^i, \hat{\boldsymbol{V}}^i \leftarrow \boldsymbol{K}^i_{\boldsymbol{C}^i} \boldsymbol{V}^i_{\boldsymbol{C}^i}$
return $\{\boldsymbol{C}^i, \hat{\boldsymbol{K}}^i, \hat{\boldsymbol{V}}^i\}$

Algorithm 2: FastGen–Token Generation.

Input: Adaptive KV cache ($\{\boldsymbol{C}^i, \hat{\boldsymbol{K}}^i, \hat{\boldsymbol{V}}^i\}$)
Output: Generated Text

$\boldsymbol{z}_0 \leftarrow$ last prompt token
for $j \in \{1, \cdots, \textit{Max Generate Length}\}$ **do**
  for *Attention Head* H_i *in LLM* **do**
    $\boldsymbol{K}^i, \boldsymbol{Q}^i, \boldsymbol{V}^i \leftarrow H_i(\boldsymbol{z}_{j-1}, \hat{\boldsymbol{K}}^i, \hat{\boldsymbol{V}}^i)$
    $\boldsymbol{K}^i_{\boldsymbol{C}^i}, \boldsymbol{V}^i_{\boldsymbol{C}^i} \leftarrow f(\boldsymbol{K}^i, \boldsymbol{V}^i, \boldsymbol{C}^i)$
    $\hat{\boldsymbol{K}}^i, \hat{\boldsymbol{V}}^i \leftarrow \boldsymbol{K}^i_{\boldsymbol{C}_i}, \boldsymbol{V}^i_{\boldsymbol{C}_i}$
  $\boldsymbol{z}_j \leftarrow$ sample from LLM prediction
return $\{\boldsymbol{z}_j\}$

$$\boldsymbol{C}^* = \arg\min_{\boldsymbol{C} \in \mathcal{C}} \text{CacheMemoryCost}(\boldsymbol{C}) \quad \text{s.t.} \quad |\boldsymbol{A} - \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T_{\boldsymbol{C}})| \leq 1 - T,$$

EXPERIMENT

Model Size	KV Cache Full	KV Cache FastGen	Pruned ratio	T	Win rate
7B	32Gb	14Gb	56.6%	91%	30.8%
		20Gb	39.8%	95%	37.7%
		27Gb	16.9%	98%	47.4%
13B	63Gb	29Gb	53.4%	91%	32.0%
		38Gb	39.0%	95%	39.9%
		51Gb	18.3%	98%	48.7%
30B	158Gb	69Gb	56.7%	93%	37.0%
		81Gb	48.8%	95%	42.5%
		115Gb	27.4%	98%	47.5%
65B	320Gb	140Gb	56.3%	93%	40.9%
		176Gb	44.9%	95%	44.2%
		205Gb	36.0%	98%	49.8%

从内存，时间分析

Generation Length	Overall Generation Duration (s)	Profiling Duration (s)	Decoding Time Per Token (s)	Profiling/Overall (%)
128	30.98	0.11	0.10	0.35%
256	50.1	0.11	0.10	0.21%
512	94.98	0.11	0.10	0.12%
1024	157.43	0.11	0.10	0.07%

Batch size	1				2			8		16
[prompt len, gen len]	[32,512]	[32,2048]	[32,8192]	[32,16384]	[512,32]	[512,512]	[4096,4096]	[512,512]	[4096,4096]	[512,512]
HF	13.35	57.37	299	799.14	1.12	19.16	167.64	23.44	OOM	OOM
DS	11.58	47.12	201.23	435.74	0.79	10.45	91.04	12.93	127.94	OOM
FastGen	11.21	44.6	179.43	359.83	0.73	9.71	76.93	10.57	82.16	OOM
Speed-up(%) over HF	16.03%	22.30%	40.00%	55.00%	34.80%	49.30%	54.10%	54.90%	-	OOM
Speed-up(%) over DS	3.20%	5.35%	10.83%	17.42%	7.59%	7.08%	15.50%	18.25%	35.78%	OOM

	Feasible Policy Set	Pruned KV Ratio	Win Rate
	$\mathcal{C}$	36.04%	49.75%
special	$\{C_{\text{punct.}}, C_{\text{punct.+frequent}}, C_{\text{punct.+frequent+local}}, C_{\text{full}}\}$	31.16%	47.64%
punct	$\{C_{\text{special}}, C_{\text{special+frequent}}, C_{\text{special+frequent+local}}, C_{\text{full}}\}$	34.23%	49.56%
local	$\{C_{\text{special}}, C_{\text{special+punct.}}, C_{\text{special+punct.+frequent}}, C_{\text{full}}\}$	30.18%	49.06%
frequent	$\{C_{\text{special}}, C_{\text{special+punct.}}, C_{\text{special+punct.+local}}, C_{\text{full}}\}$	21.26%	46.08%

frequent和special是最重要的策略，移除它们将分别导致3.67%和2.11%的胜率下降

Cache Order	Pruned KV Ratio	Win Rate
$C_{\text{special}} \rightarrow C_{\text{punct.}} \rightarrow C_{\text{frequent}} \rightarrow C_{\text{local}}$	36.04%	49.75%
$C_{\text{special}} \rightarrow C_{\text{frequent}} \rightarrow C_{\text{local}} \rightarrow C_{\text{punct.}}$	36.40%	47.64%

Table 5: Policy order ablation on fine-tuned Llama 1-65B with AlpacaEval.

EXPERIMENT

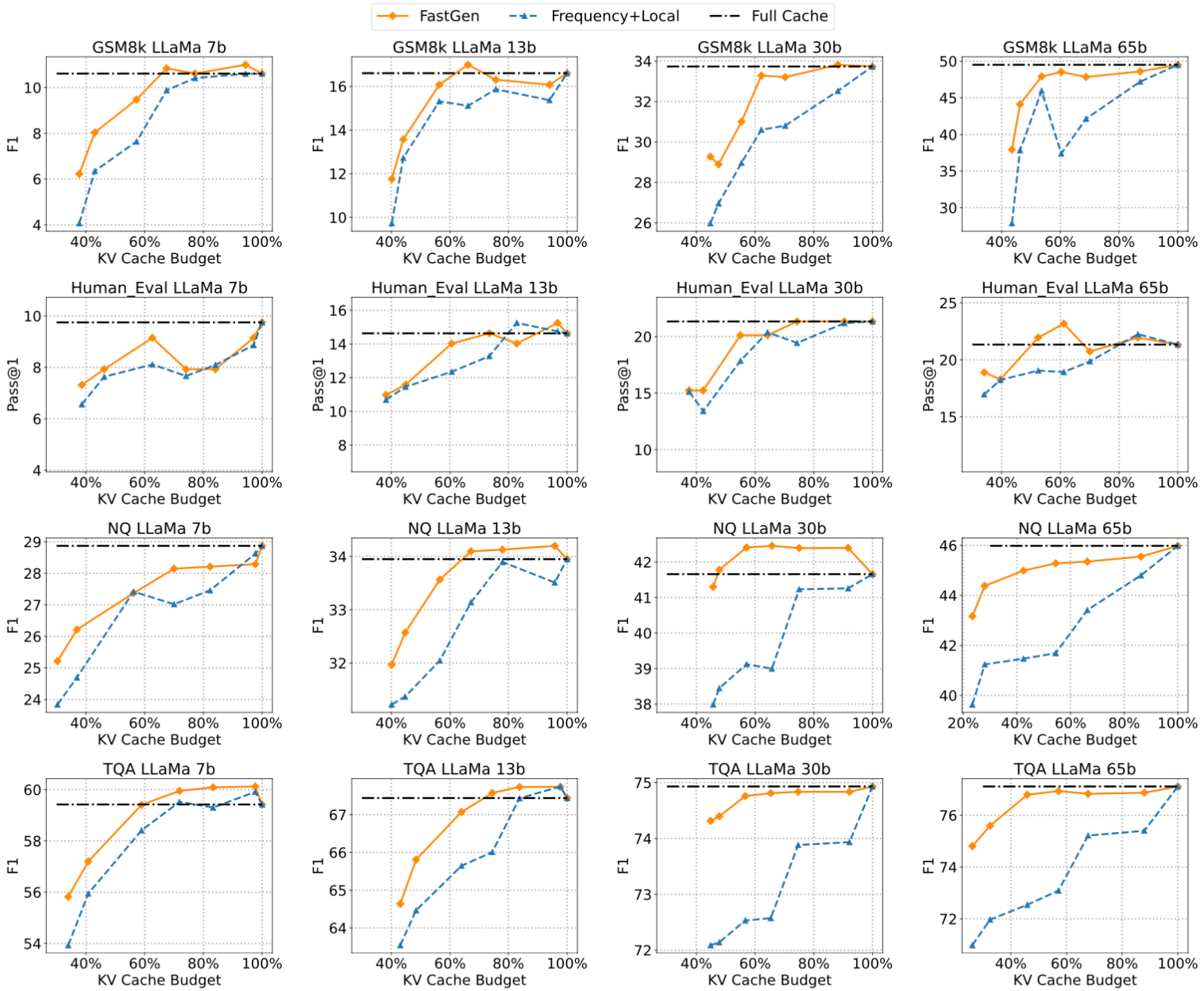


Figure 5: Performance of Adaptive KV Cache (FastGen) and Fixed KV Cache (Frequency+Local; Zhang et al., 2023 and Liu et al., 2023a) of Llama 1 on GSM8k, HumanEval, NQ, and TQA.