NEURAL INFORMATION PROCESSING SYSTEMS

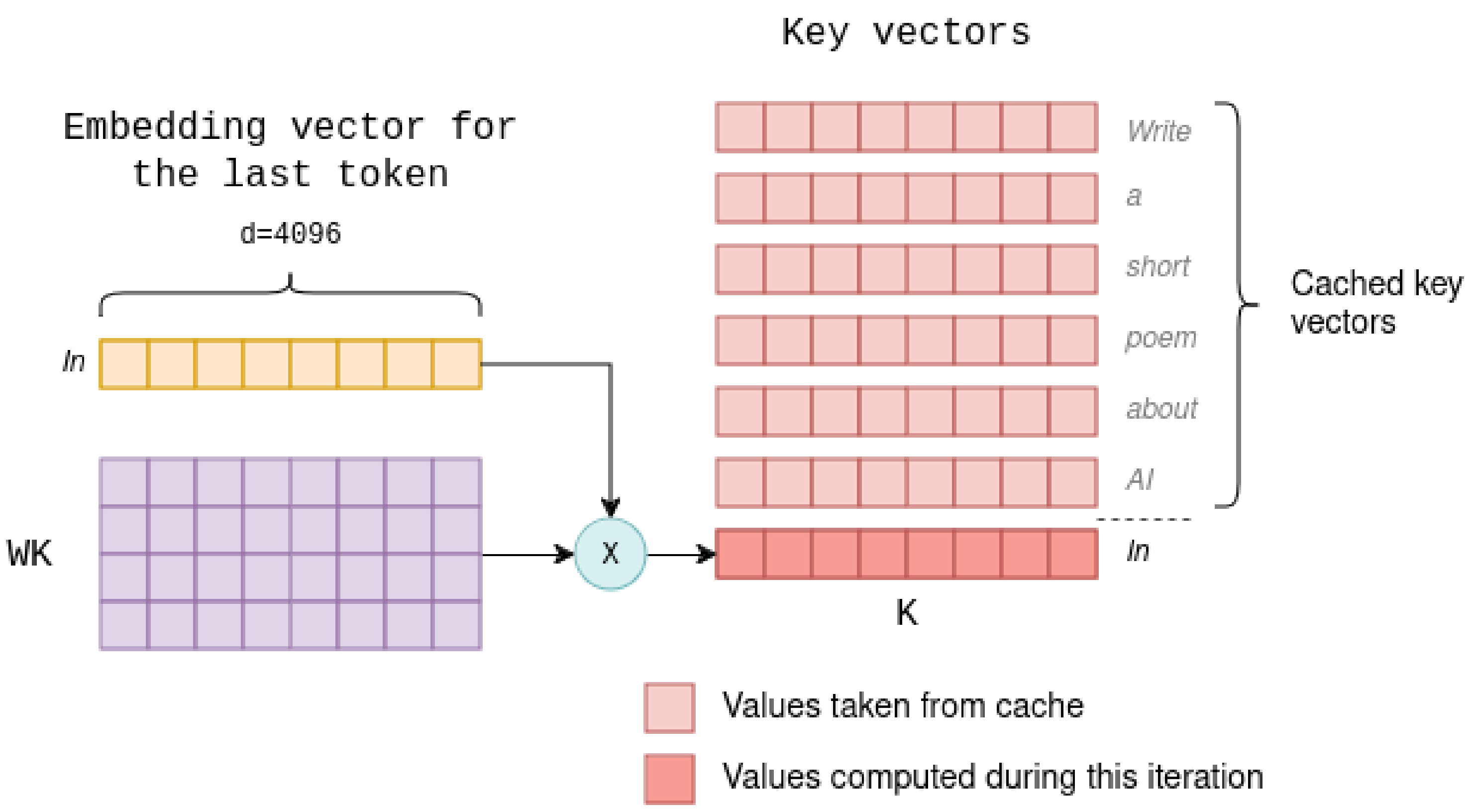
PREFIXKV: ADAPTIVE PREFIX KV CACHE IS WHAT VISION INSTRUCTION-FOLLOWING MODELS NEED FOR EFFICIENT GENERATION

Ao Wang[1], Hui Chen[2], Jiaxin Li[1], Jianchao Tan[3], Kefeng Zhang[3], Xunliang Cai[3] ,Zijia Lin[1], Jungong Han[1], Guiguang Ding[1,†]

[1]Tsinghua University [2]BNRist [3]Meituan Inc.

Background

- In prefilling, the key and value vectors are computed for all the input tokens
- In decoding, only the key and value vectors for the newly generated token are calculated



Background

- For every token, two vectors for each head and each layer are stored using FP16. The size is $2 * 2 * d_h * H * L$ bytes

Model	Cache size per token
Llama-2-7B	512KB
Llama-2-13B	800KB

- Accommodate the full context size and batch size, the result is $2 * 2 * d_h * H * L * C * B$. In Llama-2-13B, the size is 25GB under the context of 4096 tokens and batches of 8, which is similar to the model size
- Reduce the KV cache size to reduce the memory footprint and increase the inference speed

Background

- General KV cache compression framework
- Prefilling stage
  - The importance of each KV vector is derived based on the attention scores or the distance to the output
  - The most important ones are retained while the less important KV vectors are removed
- Decoding stage
  - With the inclusion of KV vectors for newly generated tokens, the importance of each KV vector is updated
  - Less critical ones are removed to ensure that the cache size consistently aligns with the overall budget

Background

- Importance estimation
  - The attention score that token t_l^n received from token t_l^m

$$\boldsymbol{a}_l^{i,m,n} = \frac{\exp(\boldsymbol{q}_l^{i,m} \cdot \boldsymbol{k}_l^{i,n})}{\sum_{j\leq m} \exp(\boldsymbol{q}_l^{i,m} \cdot \boldsymbol{k}_l^{i,j})}$$

  - Leverage the sum of attention scores as the importance

$$\boldsymbol{I}_l^n = \text{Average}_i(\sum_m \boldsymbol{a}_l^{i,m,n})$$

  - Top R_l proportion of KV vectors are retained in the l-th layer

$$\sum_{l=1}^{L} \boldsymbol{R}_l N = rLN$$

- Decoding stage
  - Prune the vectors at fixed distance to the latest generated token
  - Protect the important initial instruction and related generation

Motivation

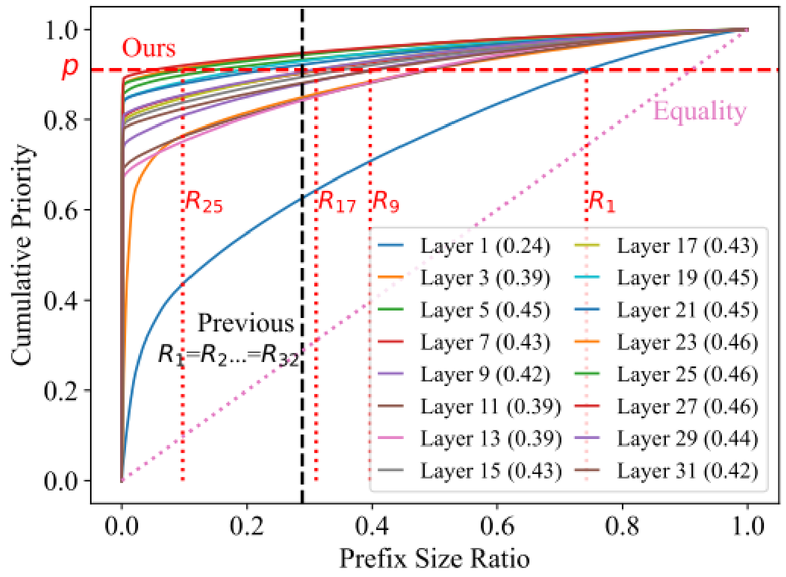
- Distinct importance distributions across layers
  - Normalize the importance metric in each layer

$$\mathcal{I}_l^n = \frac{\boldsymbol{I}_l^n}{\sum_{j=1}^{N} \boldsymbol{I}_l^j}$$

  - Sort the importance set in descending order with indices of $\{\boldsymbol{s}_l^1, \boldsymbol{s}_l^2, \ldots, \boldsymbol{s}_l^N\}$
  - Obtain the cumulative priority for each prefix size ratio o

$$\boldsymbol{P}_l^o = \sum_{j \leq oN} \mathcal{I}_l^{\boldsymbol{s}_l^j}$$

  - Analyze the Lorenz curve for the importance distribution
  - The cumulative priority growth trends vary significantly across layers
  - Previous adoption of $R_1 = R_2 = R_3 = \cdots = R_L$ suffers from contextual information loss for layers with dispersed distributions



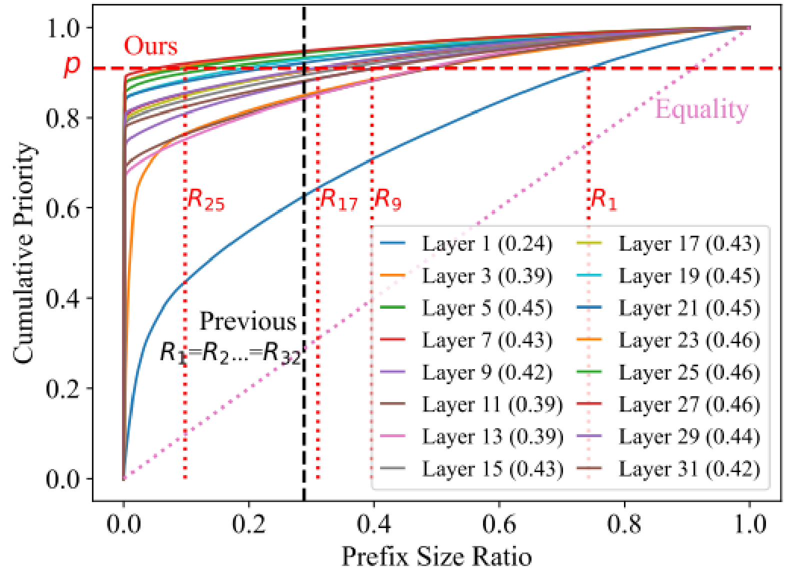
Methodology

- Global prefix configuration
  - $R_1, R_2 \dots, R_L$ constitute the prefix configuration space of the model
  - The target is to identify the optimal $R_1, R_2, \dots, R_L$ for compression
  - To reserve the information in each layer as much as possible, discover the information retention ratio p
  - The prefix size ratio R_l for the l-th layer can be derived by

    $$\boldsymbol{R}_l = \min(\{o|\boldsymbol{P}_l^o \geq p\})$$

  - The value of p needs to satisfy

    $$\sum_l \boldsymbol{R}_l = \sum_l \min(\{o|\boldsymbol{P}_l^o \geq p\}) = rL$$

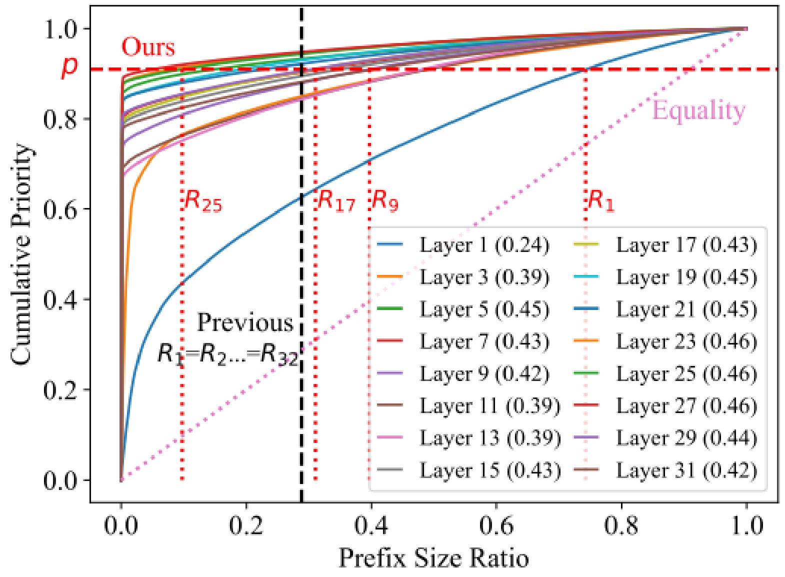


Methodology

- Binary search for optimal configuration
  - Obtain p is challenging due to its numerous possible values
  - Binary search for p to derive the prefix size ratios
  - Start with the initial interval of $[p_1, p_2]$ with $p_1 = 0$ and $p_2 = 1$
  - Try $p = \frac{p_1+p_2}{2}$ and check $\delta = \sum_l \boldsymbol{R}_l - rL$

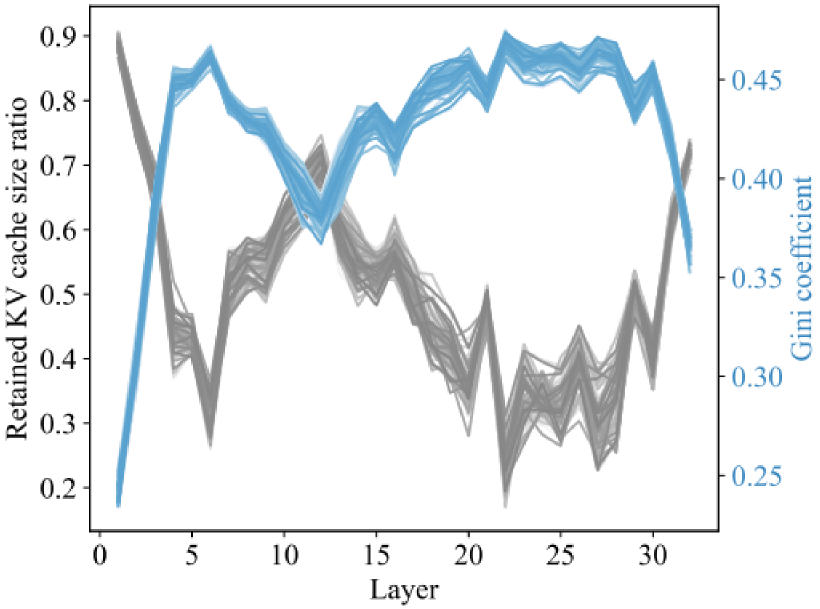
Algorithm 1: Binary Search for retention ratio p

Initialize $p_1 \leftarrow 0, p_2 \leftarrow 1$;
while $p_1 < p_2$ **do**
  $p \leftarrow \frac{p_1+p_2}{2}, \sum_l \boldsymbol{R}_l = \sum_l \min(\{o|\boldsymbol{P}_l^o \geq p\})$;
  $\delta = \sum_l \boldsymbol{R}_l - rL$;
  if $\delta == 0$ **then return** p ;
  else if $\delta < 0$ **then** $p_1 \leftarrow p$;
  else $p_2 \leftarrow p$;
return p;



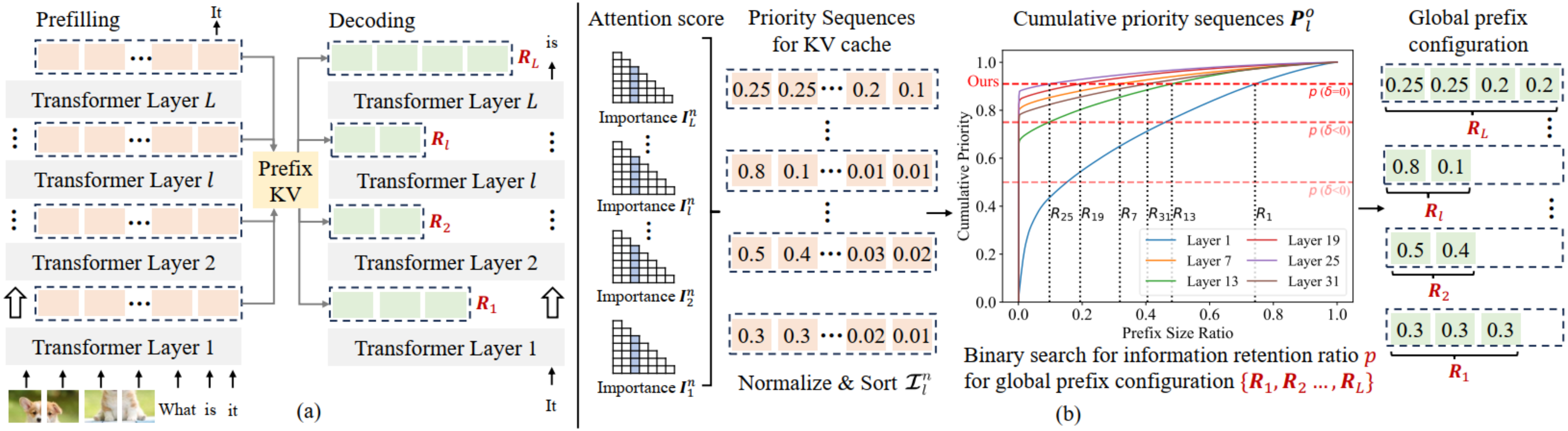
Methodology

- Offline estimation
  - Search for p introduce the extra inference overhead in the KV cache compression
  - Estimate the $R_1, R_2, \dots, R_L$ offline by the calibration data
  - It's supported by the fact that the cumulative priority sequences of layers are similar and robust across different samples



Methodology

- The overall pipeline



Experiments

- It outperforms previous works (Metrics: PPL and ROUGE score)

LLaVA-Description

Model	Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
7B	Local	66.0 / 0.22	105 / 0.14	70.0 / 0.18	47.5 / 0.17	33.8 / 0.19	14.7 / 0.30	5.50 / 0.41	4.78 / 0.50	4.03 / 0.55
	H_2O	54.5 / 0.28	48.3 / 0.31	32.0 / 0.33	18.3 / 0.32	12.9 / 0.34	7.50 / 0.41	4.28 / 0.51	4.16 / 0.53	3.72 / 0.57
	Pyramid	14.3 / 0.31	12.4 / 0.31	7.16 / 0.31	5.75 / 0.37	3.80 / 0.51	3.47 / 0.55	3.41 / 0.59	3.20 / 0.73	3.20 / 0.74
	Elastic	18.0 / 0.29	14.0 / 0.29	11.8 / 0.29	7.38 / 0.32	6.31 / 0.36	5.97 / 0.39	3.66 / 0.54	3.55 / 0.55	3.58 / 0.57
	Ours	**4.41 / 0.43**	**3.69 / 0.51**	**3.48 / 0.55**	**3.41 / 0.57**	**3.41 / 0.58**	**3.41 / 0.59**	**3.25 / 0.63**	**3.20 / 0.74**	**3.20 / 0.76**
13B	Local	60.0 / 0.15	139 / 0.12	56.3 / 0.21	16.1 / 0.27	13.2 / 0.31	7.06 / 0.37	3.72 / 0.48	3.72 / 0.52	3.25 / 0.55
	H_2O	12.4 / 0.39	10.4 / 0.39	8.50 / 0.40	4.56 / 0.46	3.78 / 0.49	3.58 / 0.49	3.16 / 0.55	3.28 / 0.57	3.06 / 0.59
	Elastic	14.9 / 0.30	5.75 / 0.35	4.41 / 0.40	3.55 / 0.50	3.36 / 0.52	3.28 / 0.53	2.97 / 0.58	2.89 / 0.60	3.02 / 0.59
	Ours	**3.72 / 0.48**	**3.17 / 0.53**	**2.97 / 0.59**	**2.92 / 0.60**	**2.89 / 0.60**	**2.84 / 0.61**	**2.77 / 0.69**	**2.73 / 0.74**	**2.73 / 0.79**

MMVet

Model	Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
7B	Local	109 / 0.11	90.0 / 0.08	99.0 / 0.13	99.0 / 0.16	66.0 / 0.16	28.4 / 0.27	12.4 / 0.34	7.88 / 0.41	6.28 / 0.46
	H_2O	158 / 0.25	120 / 0.26	72.5 / 0.29	35.3 / 0.31	18.6 / 0.30	10.3 / 0.39	7.09 / 0.44	6.22 / 0.46	5.72 / 0.49
	Pyramid	20.8 / 0.26	10.4 / 0.28	7.50 / 0.30	5.75 / 0.34	5.63 / 0.46	5.50 / 0.46	5.41 / 0.53	5.28 / 0.73	5.28 / 0.75
	Elastic	40.5 / 0.25	21.0 / 0.25	14.9 / 0.29	11.3 / 0.29	9.06 / 0.32	7.63 / 0.38	5.97 / 0.46	5.56 / 0.48	5.53 / 0.54
	Ours	**7.38 / 0.39**	**5.97 / 0.41**	**5.72 / 0.46**	**5.53 / 0.46**	**5.50 / 0.48**	**5.44 / 0.50**	**5.38 / 0.59**	**5.28 / 0.74**	**5.28 / 0.77**
13B	Local	135 / 0.15	120 / 0.14	77.0 / 0.24	53.8 / 0.26	40.5 / 0.27	18.0 / 0.34	9.06 / 0.42	6.63 / 0.39	5.41 / 0.43
	H_2O	31.6 / 0.36	30.6 / 0.38	20.8 / 0.40	10.6 / 0.43	7.75 / 0.39	6.28 / 0.44	5.63 / 0.46	5.25 / 0.47	4.88 / 0.56
	Elastic	34.3 / 0.28	11.6 / 0.34	8.00 / 0.37	6.31 / 0.44	5.81 / 0.44	5.44 / 0.49	4.97 / 0.52	4.81 / 0.51	4.81 / 0.56
	Ours	**6.28 / 0.40**	**5.16 / 0.46**	**4.88 / 0.52**	**4.78 / 0.52**	**4.72 / 0.55**	**4.72 / 0.57**	**4.72 / 0.64**	**4.69 / 0.75**	**4.72 / 0.79**

Experiments

- The effectiveness of PrefixKV

Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
Baseline	41.8	26.6	20.4	15.4	11.8	9.06	6.47	5.75	5.72
Pyramid.	20.8	10.4	7.50	5.75	5.63	5.50	5.41	**5.28**	**5.28**
PrefixKV	**7.38**	**5.97**	**5.72**	**5.53**	**5.50**	**5.44**	**5.38**	**5.28**	**5.28**

- The inference efficiency of PrefixKV

Batch Size	Model Size	Token Length	Latency (s)		Throughput (token/s)	
			PrefixKV	Full Cache	PrefixKV	Full Cache
8	13B	1024+512	20.0 / 24.3 / 27.5 / 29.7	30.5	204.6 / 168.1 / 148.7 / 137.6	134.1
16	13B	624+256	11.7 / 14.2 / 15.9 / 17.3	17.8	349.5 / 288.0 / 256.3 / 236.5	230.2
16	7B	1024+512	16.8 / 22.5 / 26.6 / 29.5	30.7	486.7 / 363.3 / 307.9 / 276.9	266.6
48	7B	624+256	13.1 / OOM / OOM / OOM	OOM	934.4 / OOM / OOM / OOM	OOM

Experiments

- The effectiveness of offline estimation

Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
Offline	7.38	5.97	5.72	5.53	5.50	5.44	5.38	5.28	5.28
Online	7.38	5.97	5.66	5.53	5.50	5.41	5.38	5.28	5.28

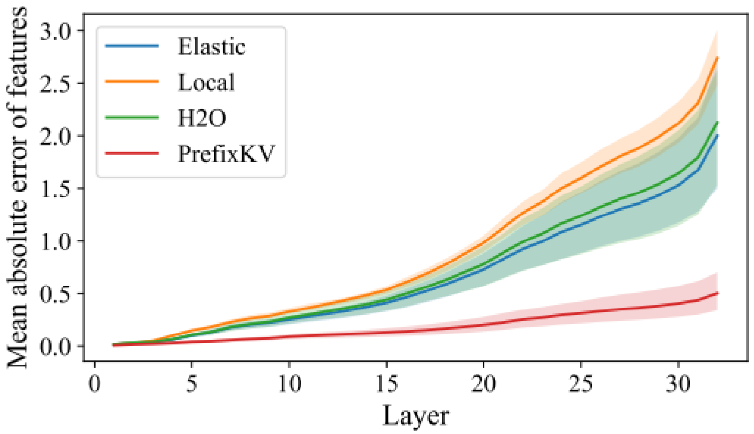
- The impact of sample size

Number	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	7.63	6.03	5.72	5.53	5.53	5.44	5.38	5.28	5.28
5	7.50	6.03	5.72	5.53	5.50	5.41	5.38	5.28	5.28
10	7.38	5.97	5.72	5.53	5.50	5.44	5.38	5.28	5.28
20	7.38	5.97	5.72	5.53	5.50	5.41	5.38	5.28	5.28

- Combination with merging

Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
PrefixKV	7.38	5.97	5.72	5.53	5.50	5.44	5.38	5.28	5.28
Position	7.63	6.06	5.75	5.53	5.44	5.31	5.31	5.28	5.28
Feature	7.38	5.97	5.72	5.53	5.44	5.31	5.28	5.28	5.28

- The feature disturbance



Experiments

- Chat examples

User	What do you think is going on in this snapshot?
Local	The two girls, two girls, two girls, two girls, two girls,...
H_2O	The image shows two young girls are two young girls are two young girls are two girls...
Elastic	The image shows a young girls are two young girls are playing with a dog is a small children are playing with a small children are two young girls are playing with a toy dog.
Ours	The image features two young girls standing next to each other, both holding stuffed animals. One girl is holding a teddy bear, while the other girl has a stuffed dog. They appear to be enjoying their time together.

User	What's happening in the scene?
Local	The snowyards,
H_2O	The image of a group of people are skiers are skiing down a snowy mountain slope, the image of a group of people are skiing down a snowy mountain scene of people are skiing down a snowy mountain...
Elastic	The image shows a group of people are skiers are enjoying a snowy mountain skiing in the snowy mountain scene with a group of people are skiers are skiing down a snowy mountain.
Ours	The image captures a group of people skiing on a snowy mountain slope. There are at least ten people visible in the scene, scattered across the slope, enjoying the winter sport. Some of the skiers are closer to the foreground, while others are further back, creating a sense of depth in the image. The snowy landscape and the clear blue sky make for a beautiful backdrop for the winter sports enthusiasts.

THANKS!