迈向下一代稀疏注意力设计

原理与工程实践

黄宇翔

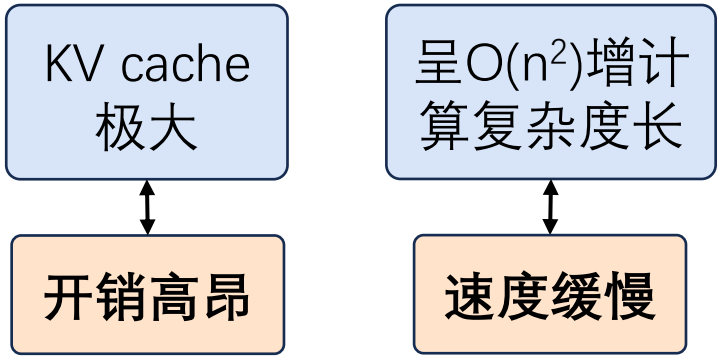
2026.03.28

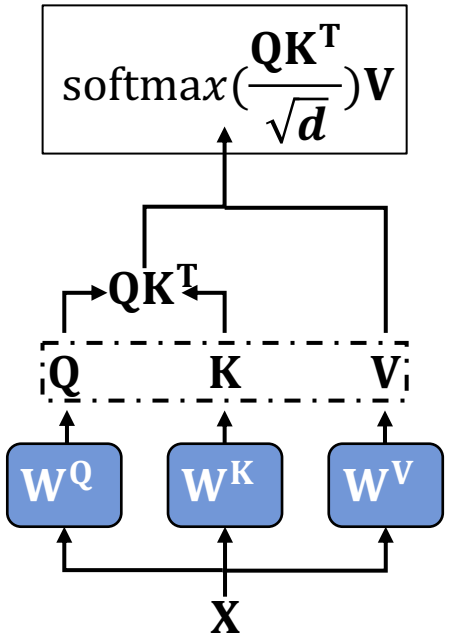
支持长上下文的挑战

- **Self-Attention是长文本推理的主要瓶颈**

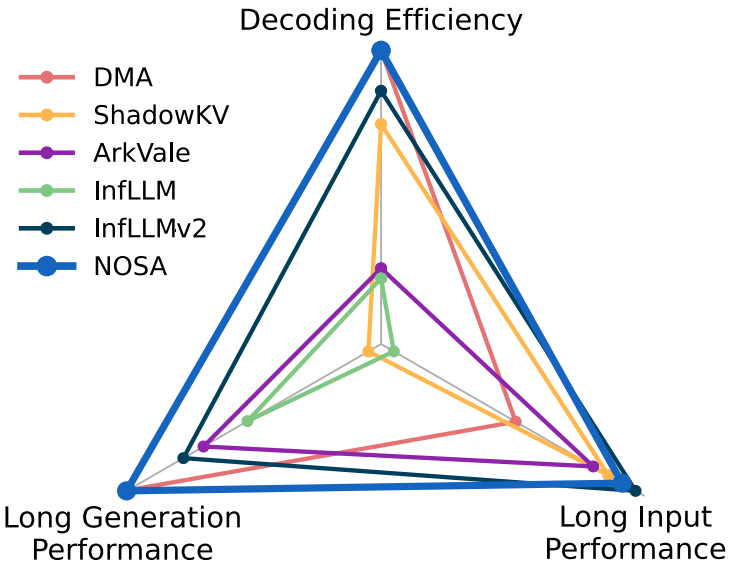
$$\mathrm{ATTN}(\mathrm{Q},\mathrm{K},\mathrm{V}) = softmax\left(\frac{\mathrm{QK^T}}{\sqrt{d}}\right)\mathrm{V}$$

- 当输入上下文变得极长时：

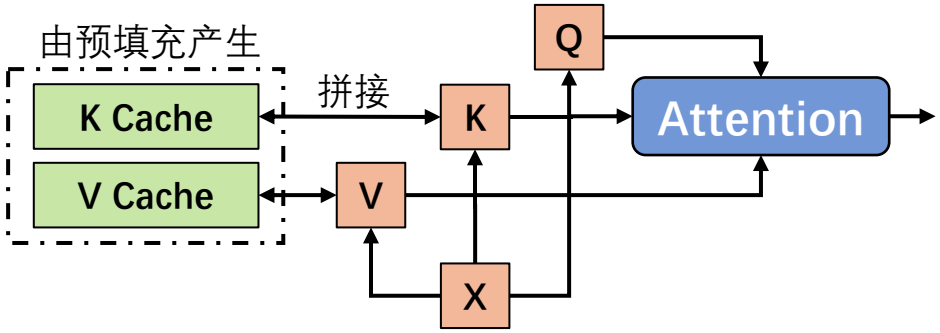




自注意力机制



- 有限的资源：计算、访存、显存容量
  - 计算与访存优化：设计适合加速的稀疏注意力
  - 显存容量：高效的KV cache管理
- 深思考时代对稀疏注意力的需求
  - 支持长输入、长输出，加速Prefill、加速Decode



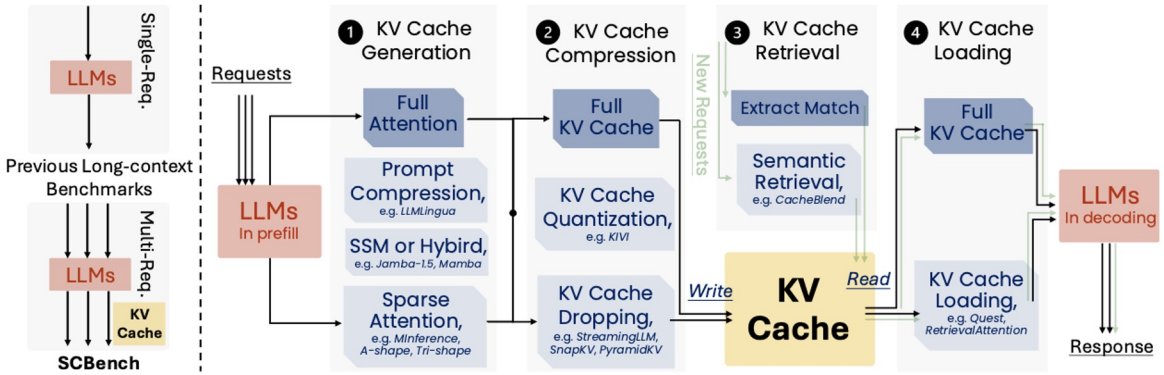
使用键值缓存解码

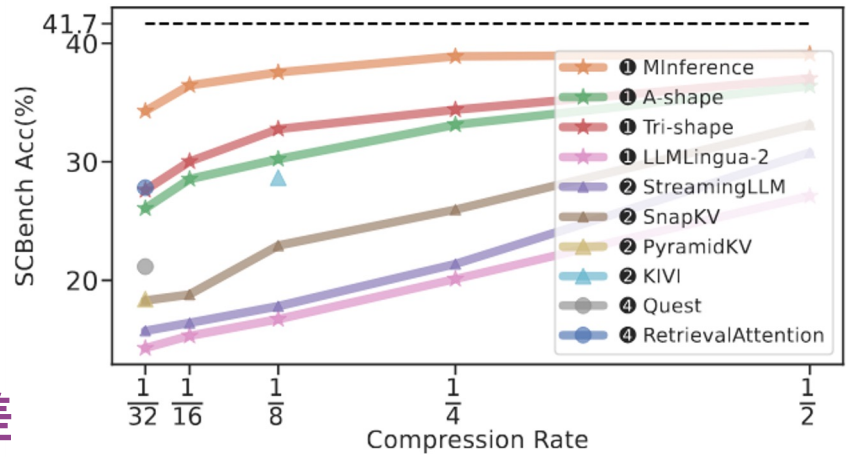
构建原生化与自适应的注意力优化方法

- KV Cache 量化/维度优化（KIVI、KVQuant等）
  - 低比特存储计算或维度压缩，压缩效果有限
- KV Cache Eviction/Offloading （SnapKV等）
  - 仅在GPU上保留重要的KV Cache
- 固定模式的稀疏注意力 （StreamingLLM等）
- 动态稀疏attention （Xattention等）
  - 通常采用块动态模式，Block Sparse Attn

无训练的方法存在训推不一致的情况，导致任务性能较差

- **原生化**的稀疏注意力：稀疏约束融入模型训练
- **自适应**的KV Cache管理：在模型架构设计/训练时考虑KV cache优化（Eviction/Offloading等）





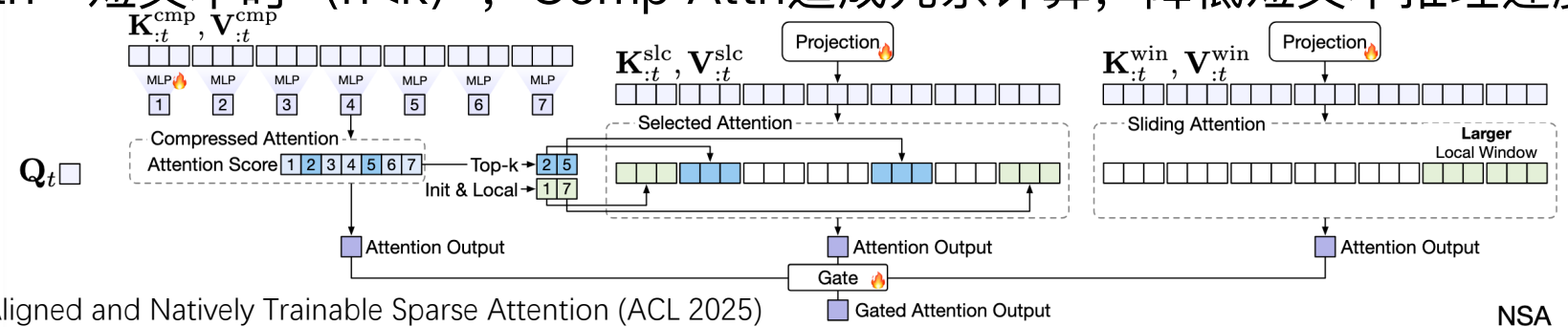
以SCBench为例，所有上述方法在压缩率较大时均产生显著的性能损失

高效长上下文处理需要可训练、原生化、自适应的优化方法

SCBench: A KV Cache-Centric Analysis of Long-Context Methods (ICLR 2025)

原生稀疏注意力

- **传统稀疏注意力：**表达能力弱，Minference等方法只能用于Prefill，训推不一致
- **训推不一致→无法做长生成/深思考**：长生成中误差逐步累计，造成生成质量崩溃
- **核心思想：**
  - 将稀疏约束引入训练，每个Query Token独立选择稀疏上下文做Attention
  - 结合GQA实现加速：用`Group Size`替代`BLOCK_SIZE_Q`满足Tensorcore GEMM尺寸
  - Deepseek NSA: 引入Compressed Attn, Block Sparse Attn, SWA三部分实现可训稀疏
- **问题：**
  - 引入新参数：难以在长文本训练阶段快速适应，容易引起训练不稳定
  - Comp Attn：短文本时（n<k），Comp Attn造成冗余计算，降低短文本推理速度



Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention (ACL 2025)

NSA

长短可切换的稀疏注意力：InfLLMv2

- **改进：**
  - Comp Attn降低短文本速度 → 去掉Comp Attn
  - Stage 1 MLP不易训练 → 简化成Mean Pooling，去掉可训参数，stage 1无需梯度
  - 将SWA, Attn Sink和Top-k选择的块放在一个kernel中计算，提升高效性
- **可训稀疏注意力**：在长文本持续预训练中使用InfLLMv2训练，8B模型仅需5B token即可
- **效果：**长文本任务、深思考任务无性能损失；稀疏训→稠密推 性能优于稠密训练

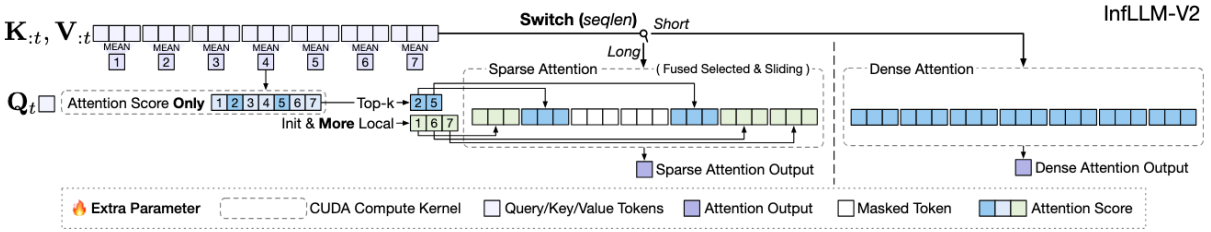


Table 1: Task Performance on RULER. Best results in sparse attention are bolded.

Method	SG1	SG2	SG3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.
FullAttn	100.00	100.00	100.00	96.00	94.00	92.00	82.00	98.50	93.20	44.40	91.33	48.00	56.00	84.26
Short+YaRN	98.00	68.00	50.00	46.00	6.00	0.00	32.00	31.50	36.00	21.40	87.33	26.00	26.00	40.63
InfLLM	98.00	6.00	4.00	10.00	10.00	10.00	9.00	7.50	70.00	16.00	80.67	18.00	24.00	27.94
MInference	**100.00**	**100.00**	**100.00**	76.00	36.00	46.00	79.50	93.50	88.00	**64.20**	**92.67**	32.00	**44.00**	73.22
NSA	**100.00**	88.00	82.00	54.00	38.00	30.00	59.00	61.50	56.00	34.40	86.00	56.00	34.00	59.92
InfLLM-V2 (Sparse)														
w/ *LSE Approx*	**100.00**	**100.00**	**100.00**	**94.00**	**82.00**	62.00	**98.50**	94.50	**98.00**	50.40	82.67	**72.00**	40.00	**82.62**
w/o *LSE Approx*	**100.00**	**100.00**	**100.00**	92.00	80.00	**64.00**	**98.50**	**95.50**	**98.00**	47.80	81.33	70.00	40.00	82.09
InfLLM-V2 (Dense)	100.00	100.00	100.00	94.00	98.00	98.00	99.00	98.00	98.40	52.80	90.00	76.00	44.00	88.32

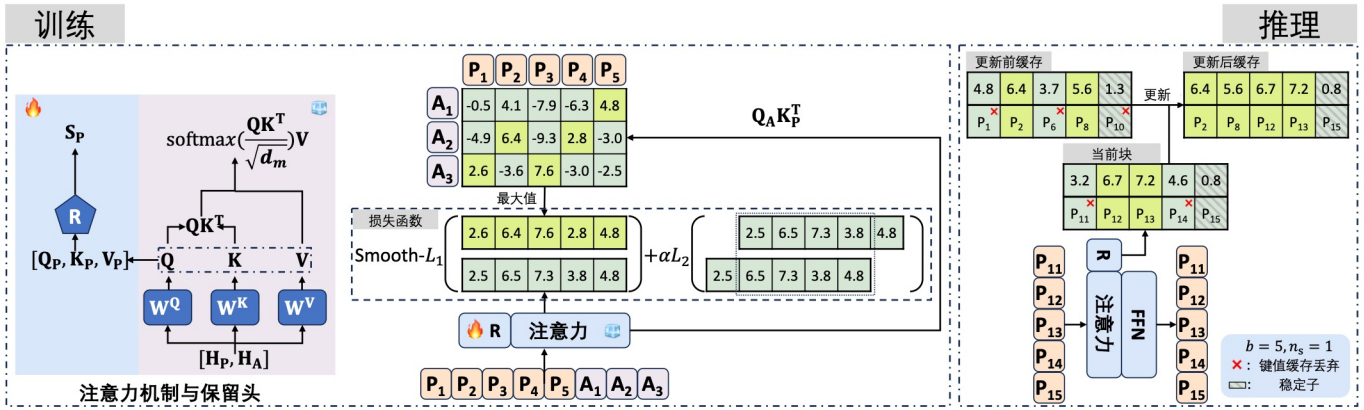
Table 3: Task Performance on Long Reasoning Tasks.

Method	MATH-500	AIME 24	AIME 25	LCB v5	LCB v6	Avg. ↑
FullAttn	86.00	37.50	30.63	30.67	29.14	42.79
NSA	83.80	28.75	23.54	25.15	25.14	37.28
InfLLM-V2 (Sparse)	87.80	38.33	29.38	29.94	27.83	42.66
InfLLM-V2 (Dense)	86.40	36.67	23.33	29.94	26.29	40.53

InfLLM-V2: Dense-Sparse Switchable Attention for Seamless Short-to-Long Adaptation (ICLR 2026)

自适应的KV Cache管理

- **传统KV Cache优化：**无需训练，依赖手工设计压缩/量化/驱逐/卸载的特征
- **缺点：**效果不佳，模型无法自行管理KV cache → 设计可训的KV cache管理方式
- **核心：**KV cache的重要性评估函数，根据KV cache的内容/语义决定其是否重要
- **方法Locret：长Prefill时，单层完整KV cache过大**
  - 使用KV cache重要性分数，与chunked prefill结合，实现流式Prefill，开销固定
  - 使用Retaining Head预测KV的重要性，在长文本QA数据上训练R Head



训练目标：

$$\underset{\mathbf{W}_1^{(i)},\mathbf{W}_2^{(i)},i=1,2\cdots,L}{\operatorname{argmin}} \mathbb{E}_{d\in\mathcal{D}}\left[\sum_{i=1}^{L}\sum_{j=1}^{h}\sum_{k=1}^{n_q(d)}\mathcal{L}\left(\tilde{\mathbf{S}}[k]_j^{(i)},\mathbf{S}[k]_j^{(i)}\right)\right]$$

损失函数：

$$\mathcal{L}\left(\tilde{\mathbf{S}}[k]_j^{(i)},\mathbf{S}[k]_j^{(i)}\right)=\text{Smooth-}\mathcal{L}_1\left(\tilde{\mathbf{S}}[k]_j^{(i)},\mathbf{S}[k]_j^{(i)}\right)+\alpha\mathcal{L}_2\left(\tilde{\mathbf{S}}[k]_j^{(i)},\tilde{\mathbf{S}}[k+1]_j^{(i)}\right)$$

Locret: Enhancing Eviction in Long-Context LLM Inference with Trained Retaining Heads on Consumer-Grade Devices (TMLR 2025)

自适应的KV Cache管理

- **Locret在低显存场景取得最佳的任务性能**
- 端侧芯片上实现流式处理长上下文
- Retaining Head：
  - 轻量化的MLP，以回归任务训练
  - 输入KV对，输出重要性分数（attn分数的估计）
- 该方法也可应用到云端推理
- 序列并行+稀疏注意力：APB
- 每个GPU只传输重要的KV Cache
- 使用 R Head 判断KV对重要性
- 相较NVIDIA StarAttn在RULER上无损加速1.6倍

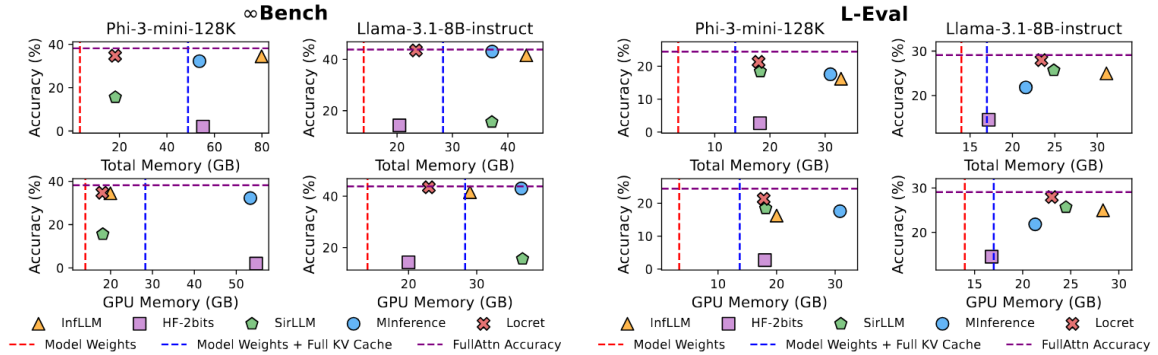
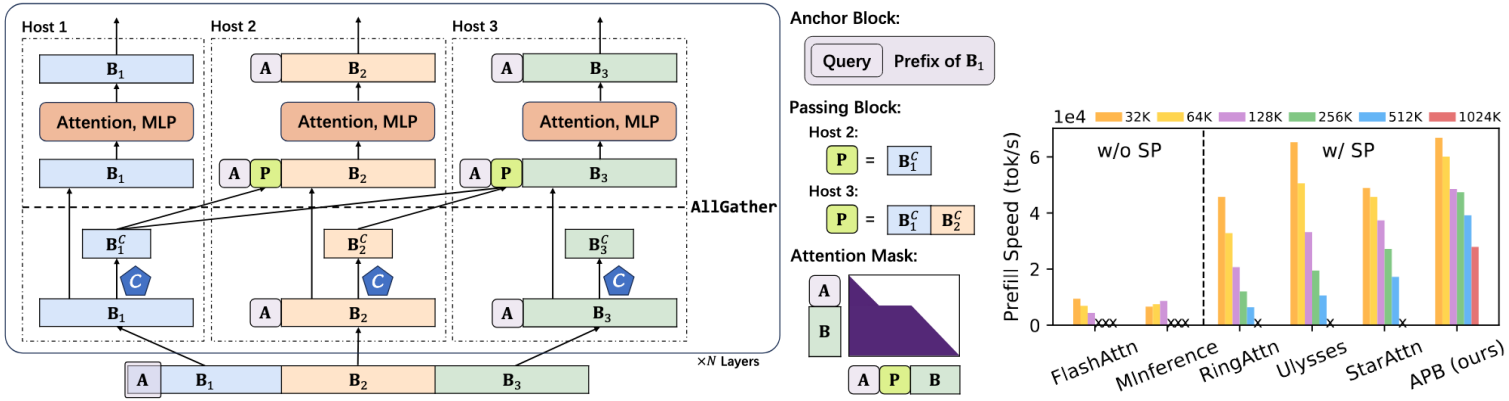


Figure 4: Memory Statistics vs. Task Performance. The red lines correspond to the theoretical size of the model weights, while the blue lines represent the total size of the model weights and the full KV cache without any compression. The purple lines indicate the accuracies of FullAttn. "Total Memory" represents the total memory usage of both GPU and CPU.

Table 4: Executing R.PassKey on an NVIDIA 4090. "tok/s" represents the inference speed, "C.Len" stands for the context length after truncation, and "Acc." represents task accuracy. The highest score among 128K context is marked in **bold**.

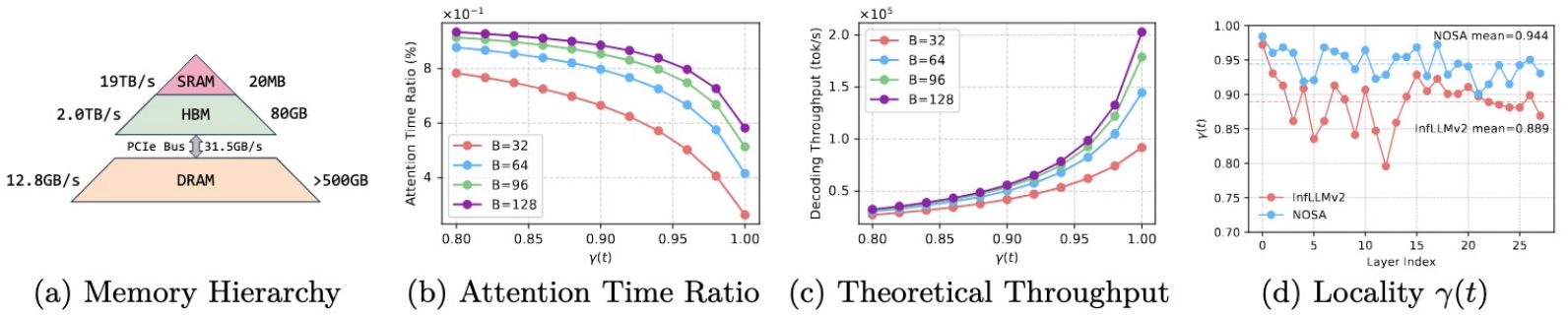
Method		FullAttn	InfLLM	HF-2bits	SirLLM	MInference	Locret	HF-2bits*	MInference*
Phi-3-mini-128K	tok/s↑	-	2276.38	-	2352.20	-	**5080.85**	1098.51	4099.92
	C.Len.↑	128K	128K	128K	128K	128K	**128K**	30K	14K
	Acc.↑	OOM	99.83	OOM	1.69	OOM	**100.00**	0.00	13.56
Llama-3.1-8B-instruct	tok/s↑	-	2287.66	1365.51	1589.75	-	**3209.10**	3680.06	5135.74
	C.Len.↑	128K	128K	128K	128K	128K	**128K**	30K	25K
	Acc.↑	OOM	100.00	35.59	1.69	OOM	**100.00**	26.78	20.34



Locret: Enhancing Eviction in Long-Context LLM Inference with Trained Retaining Heads on Consumer-Grade Devices (TMLR 2025)
APB: Accelerating Distributed Long-Context Inference by Passing Compressed Context Blocks across GPUs (ACL 2025)

合二为一：原生可卸载KV cache的稀疏注意力

- **可训稀疏/KV cache优化成为了近期主流思路**
- **现存问题：**稀疏注意力不减少KV cache，KV cache优化仍然是post training的
  - 云端serving的时候，不利于增大batch size以进一步缓解memory bound
- **核心思路：**
  - 将可训稀疏注意力应用到KV cache offloading系统中，将不激活的KV存储在CPU上
  - 训练时引入局部性约束，使Offloading的PCIe通信不至于过多
- **现象A：可训稀疏注意力InfLLMv2本身具有很好的局部性：相邻token选择KV块重叠80%**
- **现象B：80%的局部性并不够，PCIe上通信的时间仍然是主要瓶颈**



(a) Memory Hierarchy (b) Attention Time Ratio (c) Theoretical Throughput (d) Locality $\gamma(t)$

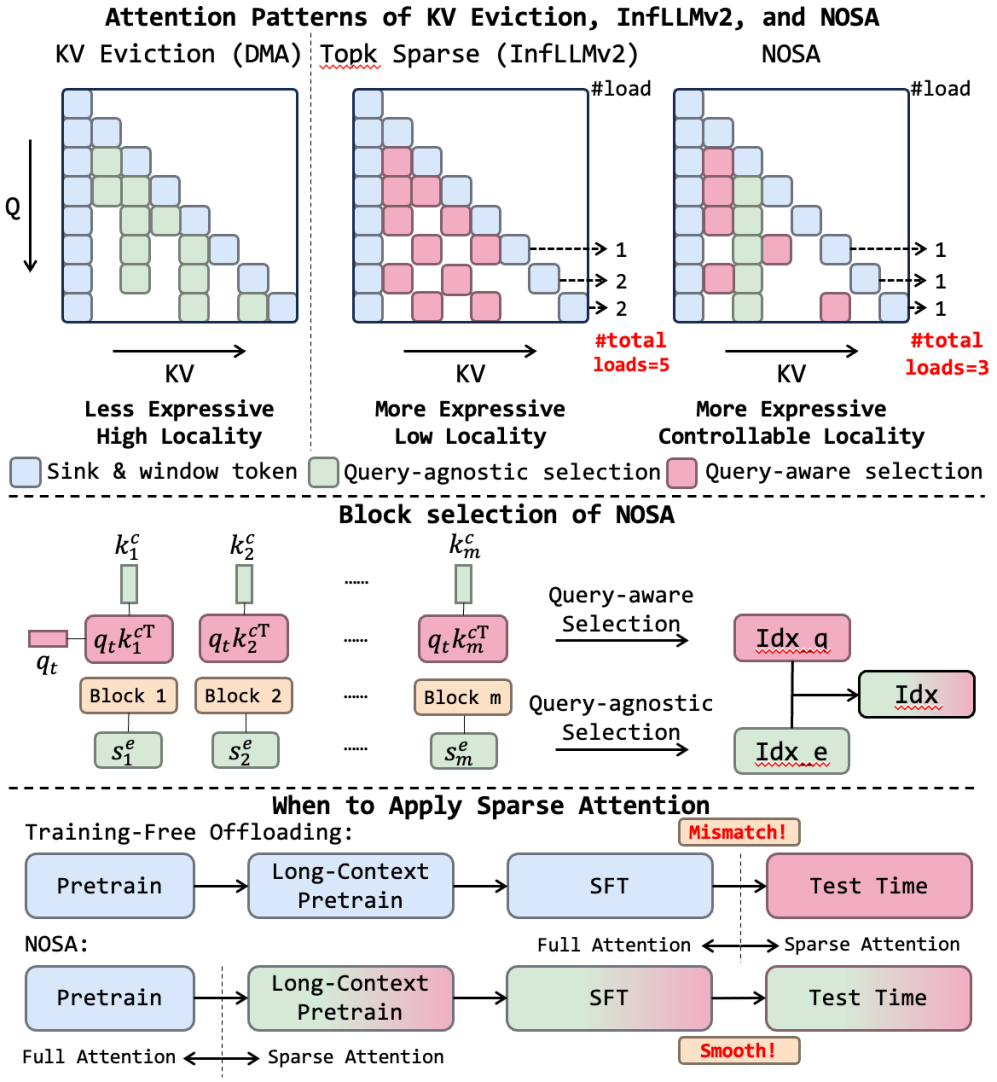
NOSA: Native, Offloadable Sparse Attention (Arxiv:2510.13602)

NOSA：原生可卸载KV cache的稀疏注意力

- 在训练中添加局部性约束
- 将KV块的选择分为：查询相关、查询无关
- 约束查询相关的选择，使用KV驱逐模式选择查询无关的token KV
- 限制查询相关选择的比例，即可对局部性产生一个下界约束

$$s_j^e = \tau(\mathbf{v}_j \mathbf{W}_1)\mathbf{W}_2,$$

$$s_{i[j]} = \begin{cases} +\infty, & s_{[j]}^q \in \mathrm{Top}_{k_q/n_b}\left(\{s_{[j]}^q\}_{j=1}^{\lfloor i/n_b \rfloor}\right); \\ s_{[j]}^e, & \text{otherwise.} \end{cases}$$



NOSA: Native, Offloadable Sparse Attention (Arxiv:2510.13602)

小结：构建原生化与自适应的注意力优化方法

- **有限的资源**：计算、访存、显存容量
  - 计算与访存优化：设计适合加速的稀疏注意力 → **InfLLMv2可训稀疏注意力**
  - 显存容量：高效的KV cache管理 → **Locret (Retaining Head), APB**
- **深思考时代对稀疏注意力的需求**
  - 支持长输入、长输出，加速Prefill、加速Decode
  - 需要能够压缩或管理KV Cache → **NOSA原生KV cache卸载方法**
- **注意力优化领域的开放问题**

A. Block-wise or Element-wise?

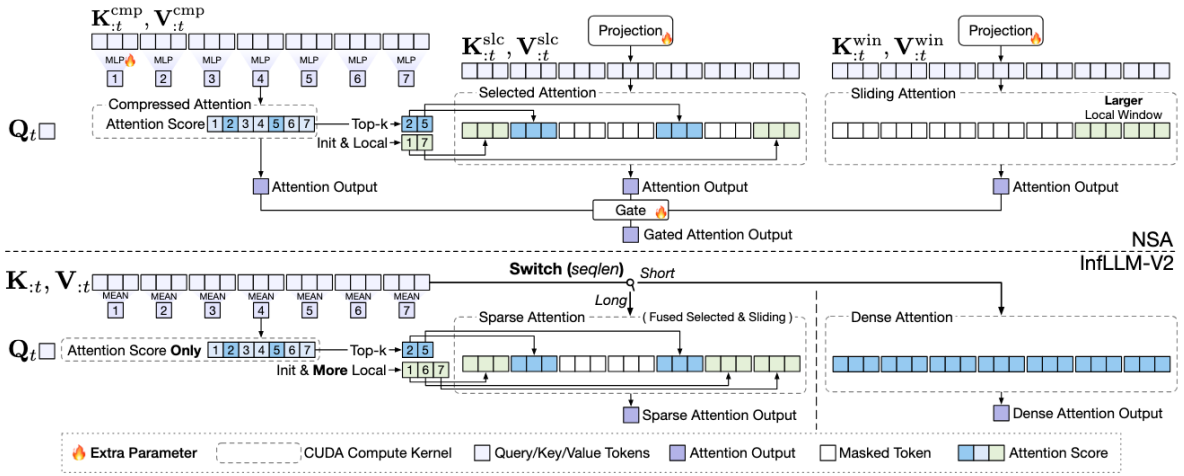
B. 有趣的现象与可解释性问题

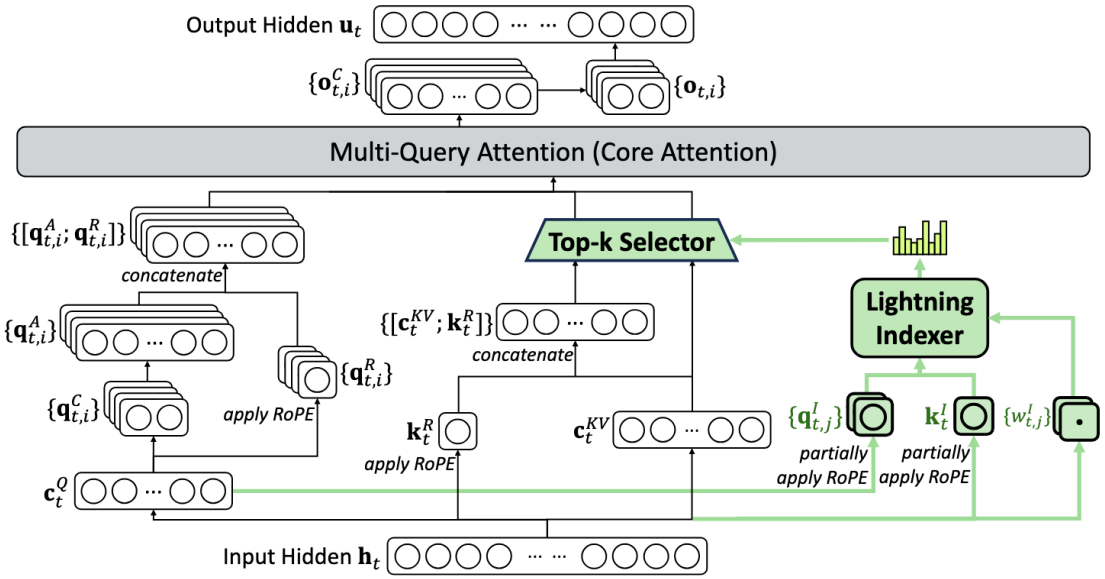
C. Top-K稀疏是最好的稀疏吗?

A. Block-wise or Element-wise?

- Deepseek NSA or DSA?
- NSA类方法：
  - 分块检索，取Top-K
  - GEMM Shape: (G, D) * (D, BK)
  - 要求 G>=16
  - 主流模型中仅有MiniCPM采用该架构
- DSA类方法
  - 有一个额外的Indexer，token级检索
  - Indexer通过KL Divergence训练
  - 与MLA紧密结合
  - 要求模型较大，且具有MLA架构

更快 vs. 更灵活：谁会占据主流?



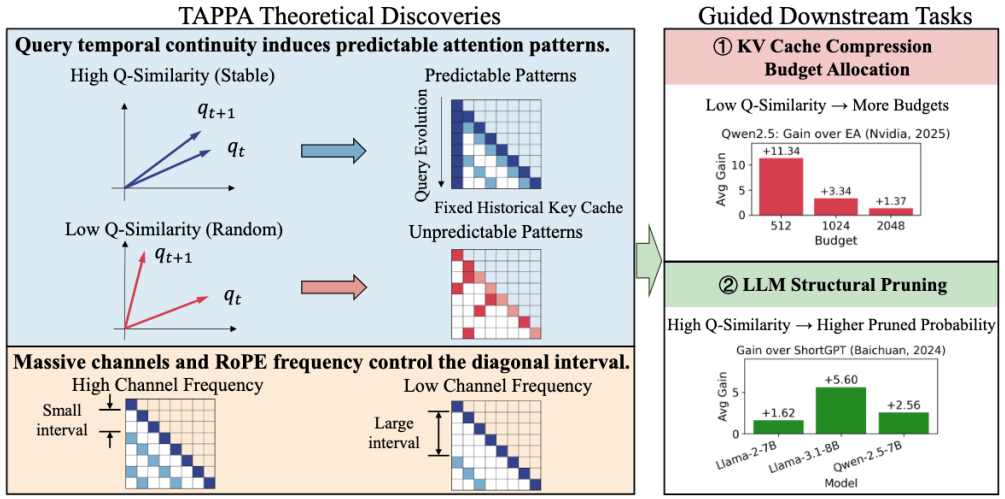


B. 有趣的现象与可解释性问题

- **有趣的现象：稀疏训→稠密推 优于 稠密训→稠密推**
- 4K激活训练>6K激活训练
- 直观解释：训练样本更难，测试样本更简单
- 真的是这样吗？为什么会出现该现象？
- **解释：a. 现象原因; b. 为什么稀疏注意力可训**
- Attn Pattern可解释性问题：RoPE模型的注意力可解释性极差
- 一些学术界给出的猜测：
  - Attn Head分化论与退化论
  - RoPE干扰论与Query相似论
  - 先验学习论

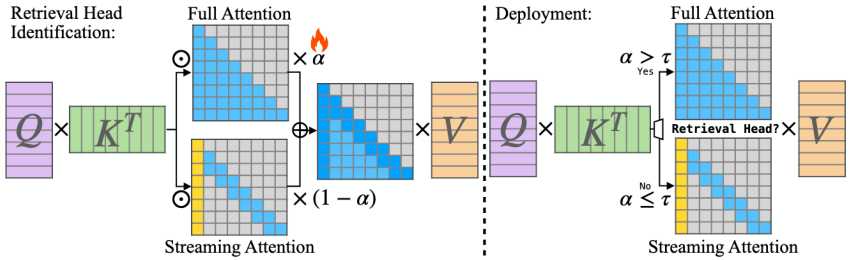
Table 2: Task Performance on LongBench and LongPPL. Best results in sparse attention are bolded.

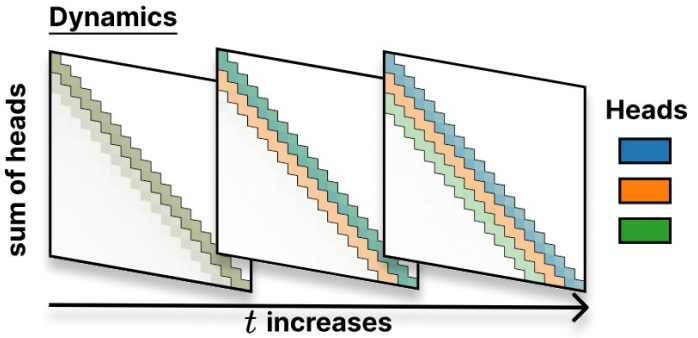
Benchmark	FullAttn	Short+YaRN	InfLLM	MInference	NSA	InfLLM-V2 (Sparse)	InfLLM-V2 (Dense)
LongBench ↑	42.30	37.86	32.30	41.55	37.10	**42.54**	42.49
LongPPL ↓	2.06	5.28	12.01	2.62	4.24	**2.12**	2.00

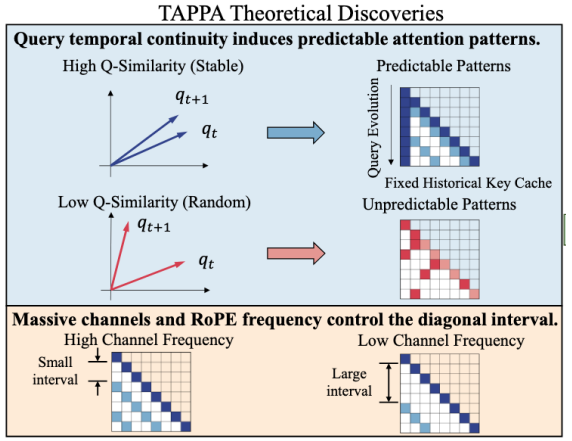


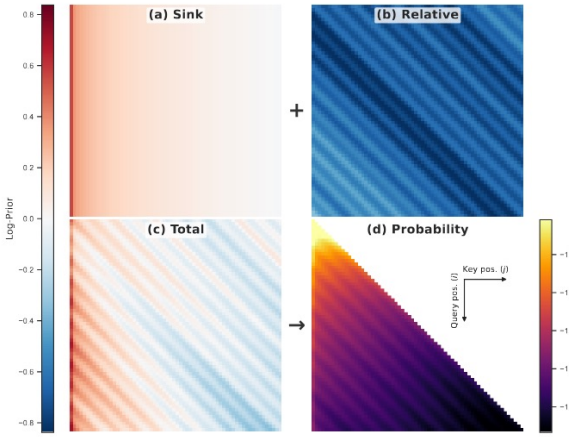
B. 有趣的现象与可解释性问题

- Attn Head分化论与退化论
  - Duo Attention将注意力头分为两类：检索头、非检索头，部分头不参与长文本建模
  - Yüksel, et. al. 提出训练过程中注意力头会逐渐退化成滑动窗口
- RoPE干扰论与Query相似论: RoPE信号过强
  - 相邻token的Q相似度决定了该头的注意力模式是否可以被预测
- 先验学习论 $$p_{ij} = \mathrm{softmax}_j\left(\frac{\langle \boldsymbol{q}_{c,i}, \boldsymbol{k}_{c,j}\rangle}{\sqrt{d_c}} + \mathcal{K}_{ij}\right),$$
  - 注意力模式是一种学习得到的位置先验
  - 可以被分解成不同强度的周期信号叠加









DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads; Incremental Learning of Sparse Attention Patterns in Transformers
Why Attention Patterns Exist: A Unifying Temporal Perspective Analysis; You Need Better Attention Priors

C. Top-K稀疏是最好的稀疏吗

- **Softmax具有Dispersion性质**

$$\lim_{n\to\infty} \frac{H^S(\text{softmax}(\mathbf{z}))}{\log n} = 1 \Rightarrow \text{KL}(\text{softmax}(\mathbf{z})||\mathbf{u}) = 0, \text{ where } u_j = \frac{1}{n}$$

- Dispersion不能保证概率单点收敛到u，但会令attention退化为对V的平均
- Softmax叠加Top-K依旧有Dispersion性质：局部Dispersion仍存在
- **Top-K不可导：对K和topk-indices均不可导**
  - InfLLMv1中的stage1不可训，若希望可训，则必须类似NSA加入Comp Attn
- **Top-K不可学：目前的方法无法给Top-K一个比较好的梯度**
  - InfLLMv2不给stage1梯度，NSA的Comp Attn对stage1的梯度不会优化topk的选择
- **Top-K不灵活：每层、每个注意力头选择的token数目是一定的**

C. Top-K稀疏是最好的稀疏吗

- **下一代稀疏注意力**
  - 长序列建模能力：去掉Softmax中的Dispersion性质
  - 良好的理论解释：稀疏是由模型结构决定的，具有良好的梯度结构
  - 自适应调控选择token数：在层、注意力头之间实现自适应稀疏
- **一些学术界的已有思路**
  - Softmax的来源：凸优化的解。设 $\Omega : \Delta^{n-1} \to \mathbb{R}_+ \cup \{+\infty\}$ 为一 n 维概率空间测度，那么的对偶算子

$$\Omega^*(\mathbf{z}) = \sup_{\mathbf{p} \in \Delta^{n-1}} \{\mathbf{p}^\top \mathbf{z} - \Omega(\mathbf{p})\}$$

即可将任意logits $\mathbf{z} \in \mathbb{R}^n$ 映射到一个概率向量 $\mathbf{p} \in \Delta^{n-1}$

  - 传统softmax是如下问题的解：$\Omega(\mathbf{p}) = -H^S(\mathbf{p}) = -\sum_{j=1}^{n} -p_j \log p_j$
  - 将测度换成Tsallis Entropy $\Omega(\mathbf{p}) = -H^T(\mathbf{p}) = -\sum_{j=1}^{n} \frac{1}{\alpha(\alpha-1)}(p_j - p_j^\alpha)$

即可得到entmax

$$p_j = \text{entmax}(\mathbf{z}; \alpha) = [(\alpha-1)z_j - \tau]_+^{\frac{1}{(\alpha-1)}} \geq 0, \ \tau = \tau(\mathbf{z}) \ s.t. \ \mathbf{p}^\top \mathbf{1} = 1$$

Sparse Sequence-to-Sequence Models (ACL 2019)

C. Top-K稀疏是最好的稀疏吗

- **Entmax稀疏模型**
  - 效果与表达能力不弱于传统Attention
  - 在长度外推上具有卓越的表现：解决了Dispersion问题

$$\lim_{n\to\infty} \frac{H^S(\text{entmax}(\mathbf{z};\alpha))}{\log n} \leq c < 1 \Rightarrow \text{KL}(\text{softmax}(\mathbf{z})||\mathbf{u}) >= c' > 0$$

Table 4: Retrieval performance on RULER benchmark. All models use NAPE positional encoding and were trained on 2048-token contexts. Best results are in **bold**.

	S-NIAH-1 (ID)		S-NIAH-1 (OOD)			S-NIAH-2 (ID)		S-NIAH-2 (OOD)	
Model	1K	2K	4K	8K	16K	1K	2K	4K	8K
Softmax	**100.0**	99.4	94.2	11.4	0.8	**100.0**	**100.0**	4.8	0.0
SSMax	**100.0**	99.8	99.2	92.0	75.2	99.4	99.2	64.4	14.8
Entmax	99.8	99.8	89.0	21.6	1.2	99.6	99.4	64.8	7.2
ASEntmax	99.6	**100.0**	**100.0**	**99.8**	**97.4**	99.4	99.4	**83.2**	**25.4**

- **提供良好的训练梯度**
  - 稀疏蒸馏/稀疏训练的方法仍在发展
  - 一种思路：使用Fenchel–Young共轭构造损失函数

$$\mathcal{L}(\theta; y) = \Omega(y) + \Omega^*(\theta) - \theta^\top y$$

$$\Omega^*(\mathbf{z}) = \sup_{\mathbf{p}\in\Delta^{n-1}} \{\mathbf{p}^\top\mathbf{z} - \Omega(\mathbf{p})\}$$

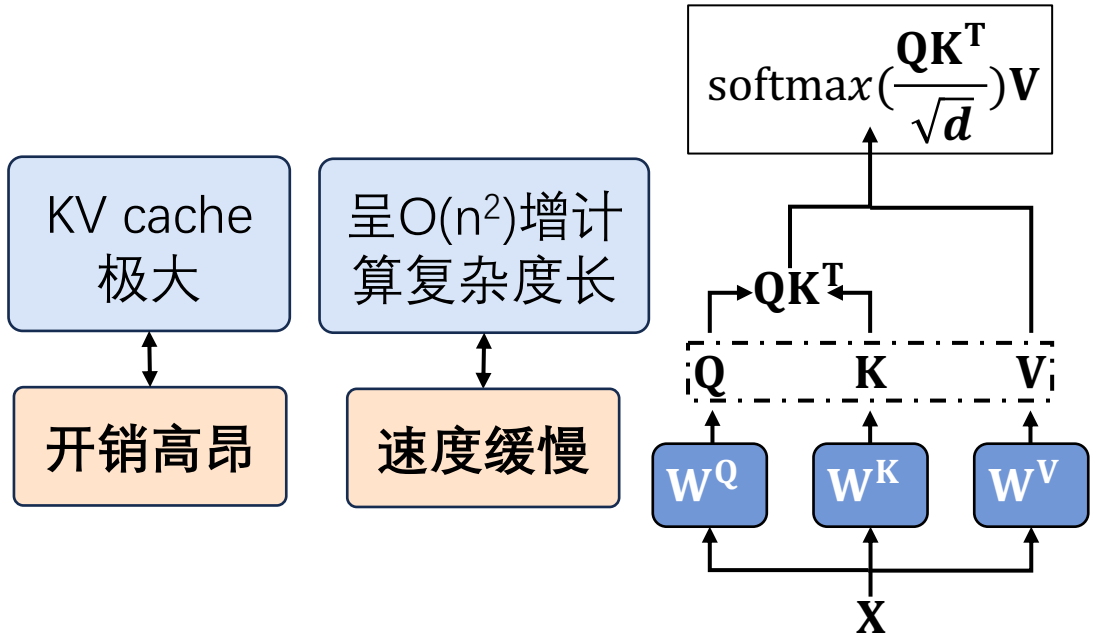
$$\Omega(y) + \Omega^*(\theta) \geq \theta^\top y$$

  - 最优化梯度条件 $\theta \in \partial\Omega^*(y)$ 给稀疏概率梯度提供了一个良好的形式

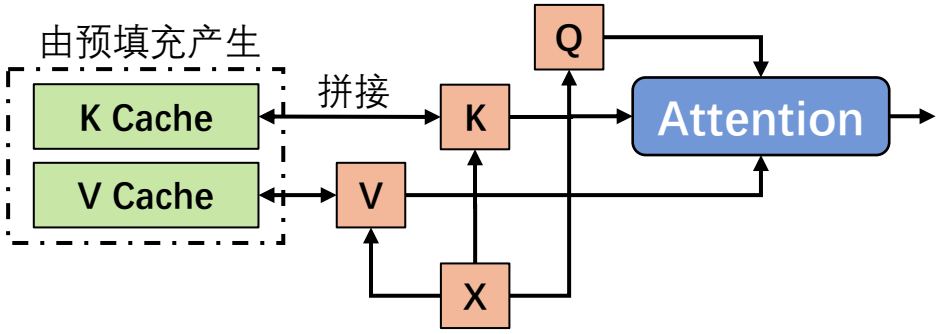
Long-Context Generalization with Sparse Attention (ICLR 2026)
Learning with Fenchel-Young Losses (JMLR 2020)

总结：稀疏化、可训稀疏化、下一代稀疏化

- 上一代注意力优化
  - 发现和使用注意力的稀疏特性
  - 方法大多无需训练
  - 软硬件协同设计
- 当前注意力优化
  - 可训稀疏：将后训练采用的优化融入预训练
  - 原生与自适应的注意力优化
- 下一代注意力优化
  - 更好的理论性质，从原理上提升attention能力
  - 更细力度的自适应稀疏注意力
  - 更好的可解释性：解释并利用稀疏性的来源



自注意力机制



使用键值缓存解码


谢谢！

迈向下一代稀疏注意力设计：原理与工程实践

黄宇翔

2026.03.28