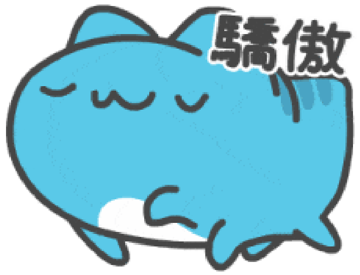
deepseek

Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention

从NSA出发，到任何地方

Native Sparse Attention

东川路第一可爱猫猫虫

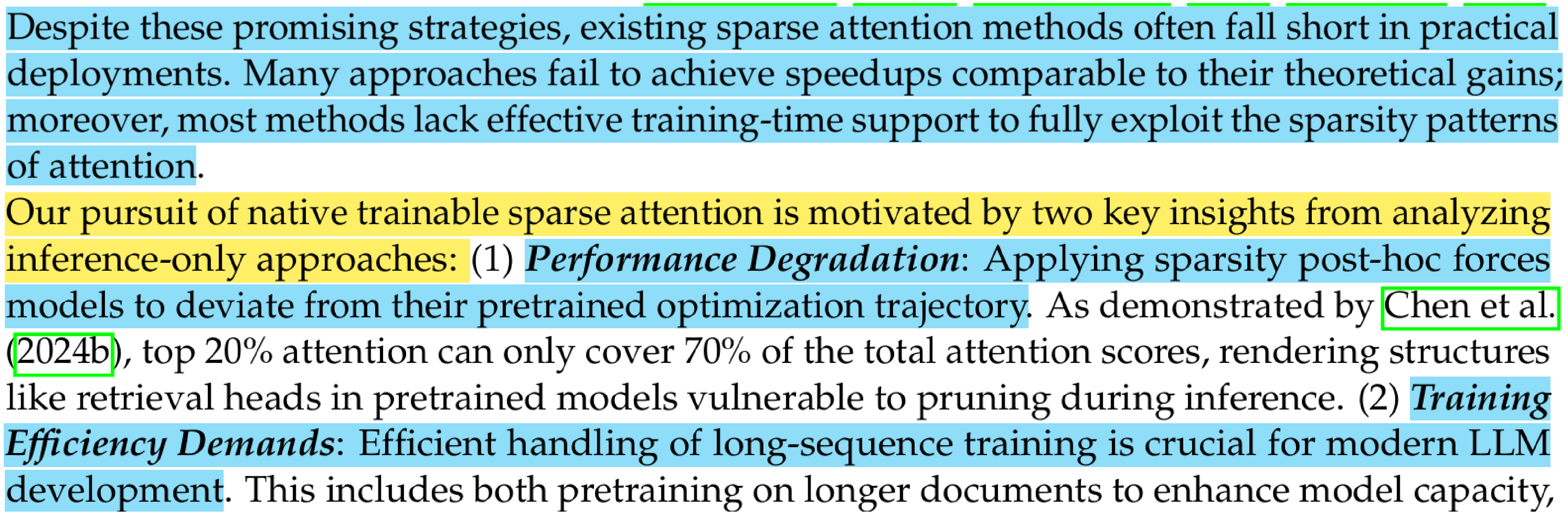


主要内容

感谢粉丝大佬
迷途小白哟
的30元档包月充电

- Native Sparse Attention产生的背景
- NSA的核心思路
  - 第一分支：粗粒度压缩
  - 第二分支：细粒度选择
  - 第三分支：滑动窗口
  - 门控机制
- 组件均可学习

Native Sparse Attention产生的背景



Despite these promising strategies, existing sparse attention methods often fall short in practical deployments. Many approaches fail to achieve speedups comparable to their theoretical gains; moreover, most methods lack effective training-time support to fully exploit the sparsity patterns of attention.

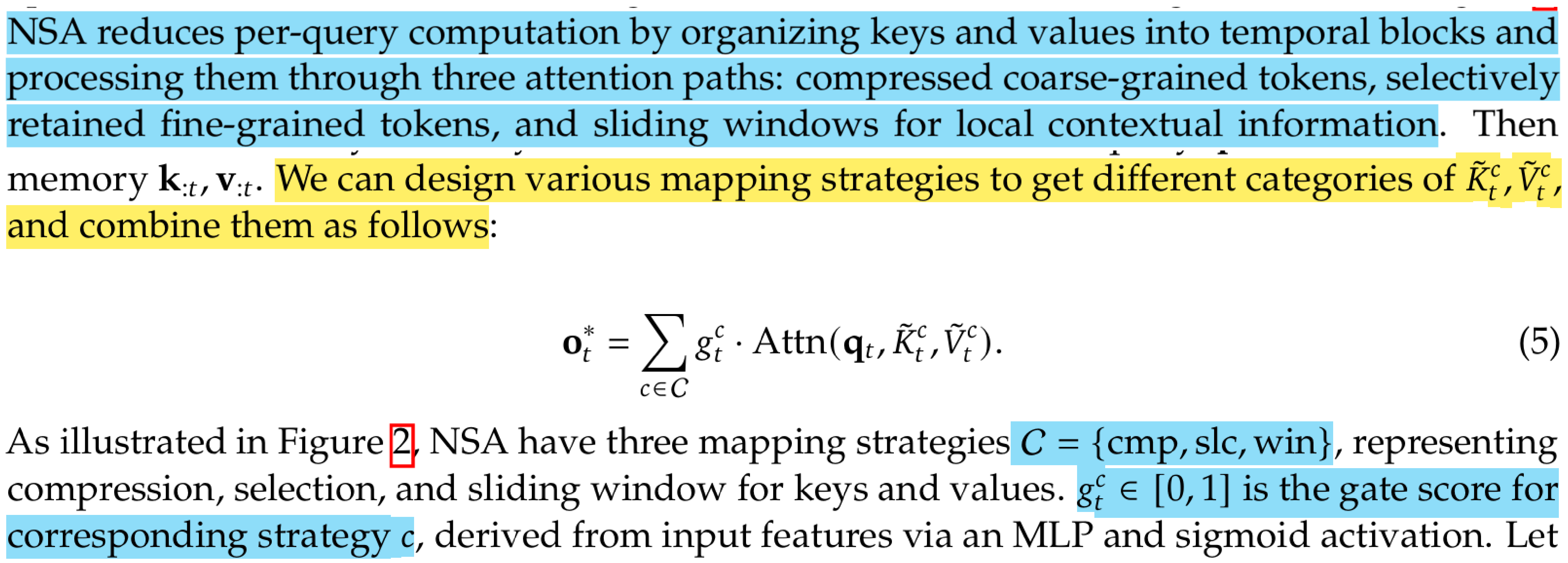
Our pursuit of native trainable sparse attention is motivated by two key insights from analyzing inference-only approaches: (1) ***Performance Degradation***: Applying sparsity post-hoc forces models to deviate from their pretrained optimization trajectory. As demonstrated by Chen et al. (2024b), top 20% attention can only cover 70% of the total attention scores, rendering structures like retrieval heads in pretrained models vulnerable to pruning during inference. (2) ***Training Efficiency Demands***: Efficient handling of long-sequence training is crucial for modern LLM development. This includes both pretraining on longer documents to enhance model capacity,

- 既有稀疏注意力的问题

  理论快，实际慢

  非原生可训练

NSA的核心思路



NSA reduces per-query computation by organizing keys and values into temporal blocks and processing them through three attention paths: compressed coarse-grained tokens, selectively retained fine-grained tokens, and sliding windows for local contextual information. Then memory $\mathbf{k}_{:t}, \mathbf{v}_{:t}$. We can design various mapping strategies to get different categories of $\tilde{K}_t^c, \tilde{V}_t^c$, and combine them as follows:

$$\mathbf{o}_t^* = \sum_{c \in C} g_t^c \cdot \mathrm{Attn}(\mathbf{q}_t, \tilde{K}_t^c, \tilde{V}_t^c). \tag{5}$$

As illustrated in Figure 2, NSA have three mapping strategies $C = \{\mathrm{cmp}, \mathrm{slc}, \mathrm{win}\}$, representing compression, selection, and sliding window for keys and values. $g_t^c \in [0, 1]$ is the gate score for corresponding strategy c, derived from input features via an MLP and sigmoid activation. Let

- 三个分支模块
- 门控机制

第一分支：粗粒度压缩

By aggregating sequential blocks of keys or values into block-level representations, we obtain compressed keys and values that capture the information of the entire block. Formally, the

resentation $\tilde{V}_t^{\text{cmp}}$. Compressed representations capture coarser-grained higher-level semantic information and reduce computational burden of attention.

$$\tilde{K}_t^{\text{cmp}} = f_K^{\text{cmp}}(\mathbf{k}_{:t}) = \left\{ \varphi(\mathbf{k}_{id+1:id+l}) \middle| 0 \leqslant i \leqslant \left\lfloor \frac{t-l}{d} \right\rfloor \right\}$$

- $\tilde{K}_t^{cmp}$：处理第t个token时，所有压缩键组成的集合
- $\tilde{K}_t^{cmp}$的形状：$d_k \times \left\lfloor \frac{t-l}{d} \right\rfloor$ d_k是key的维度 $\left\lfloor \frac{t-l}{d} \right\rfloor$是压缩块的数量
- $\tilde{K}_t^{cmp}$可以看成一个表格，每一行是一个维度，每一列是一个压缩块
- f_t^{cmp}为压缩函数，输入为$\mathrm{k}_{:t}$

  $\mathrm{k}_{:t}$表示从第 1 个到第 t 个 token 的所有原始键

第一分支：粗粒度压缩

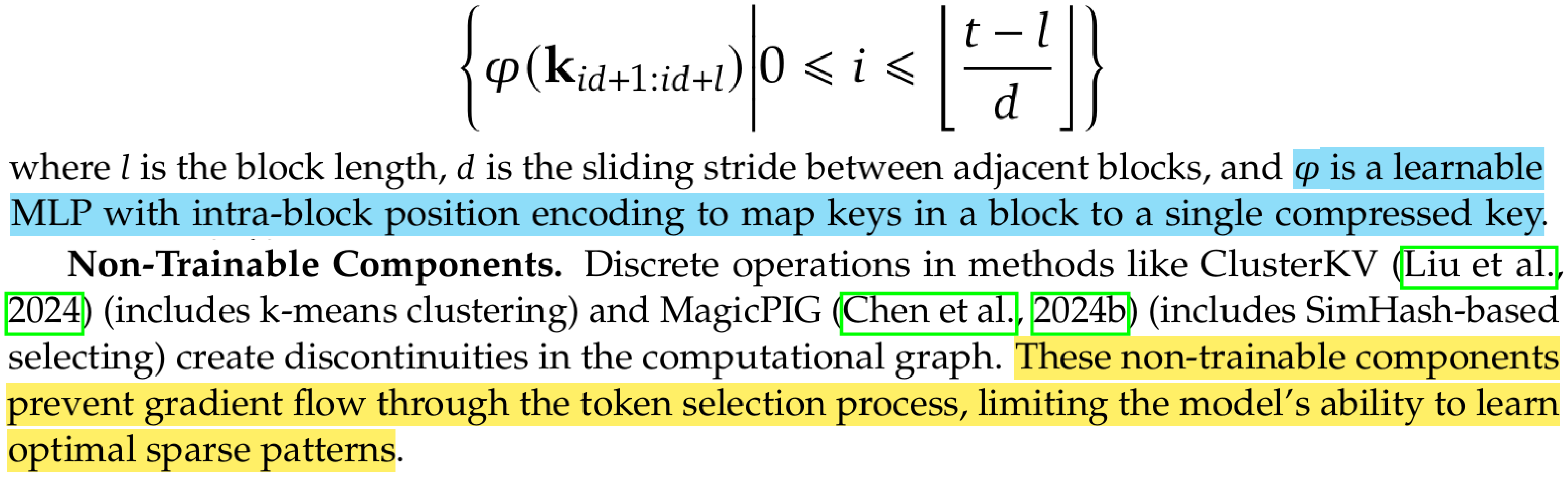
$$\left\{\varphi(\mathbf{k}_{id+1:id+l}) \middle| 0 \leqslant i \leqslant \left\lfloor \frac{t-l}{d} \right\rfloor \right\}$$

- $\left\lfloor \frac{t-l}{d} \right\rfloor$表示压缩块的总数

  要对前 t 个 token，按 l 长度、d 步长，拆成多少个块来压缩
- $id+1{:}\,id+l$表示第i个块的token范围
- 举个例子 取块长度l=32，步长d=16
- 第1000个token

  $\frac{t-l}{d} = \frac{968}{16} = 60.5 \quad \left\lfloor \frac{t-l}{d} \right\rfloor = 60$
- i=0, $id+1{:}\,id+l$即1到32；i=1, $id+1{:}\,id+l$即17到48

第一分支：粗粒度压缩



$$\left\{\varphi(\mathbf{k}_{id+1:id+l}) \middle| 0 \leqslant i \leqslant \left\lfloor \frac{t-l}{d} \right\rfloor \right\}$$

where l is the block length, d is the sliding stride between adjacent blocks, and φ is a learnable MLP with intra-block position encoding to map keys in a block to a single compressed key.

Non-Trainable Components. Discrete operations in methods like ClusterKV (Liu et al., 2024) (includes k-means clustering) and MagicPIG (Chen et al., 2024b) (includes SimHash-based selecting) create discontinuities in the computational graph. These non-trainable components prevent gradient flow through the token selection process, limiting the model's ability to learn optimal sparse patterns.

- φ

  可学习的 MLP

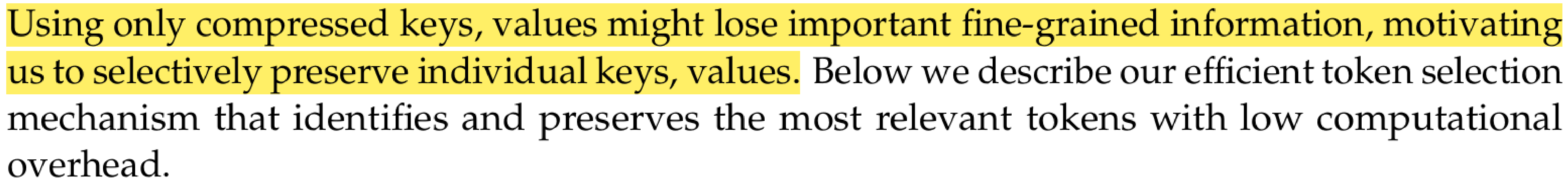
  带块内位置编码

第一分支：粗粒度压缩

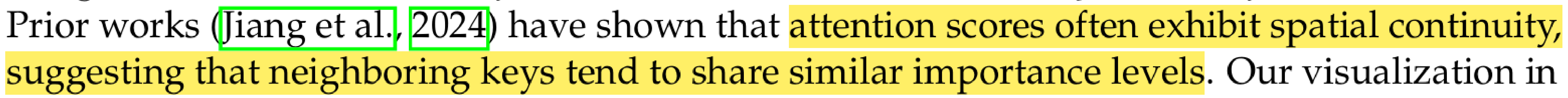
$$\tilde{K}_t^{\text{cmp}} = f_K^{\text{cmp}}(\mathbf{k}_{:t}) = \left\{ \varphi(\mathbf{k}_{id+1:id+l}) \middle| 0 \leqslant i \leqslant \left\lfloor \frac{t-l}{d} \right\rfloor \right\}$$

- 第i个块，用可学习的多层感知器，压缩key
  得到一个$\varphi(k_{id+1:id+l})$
  即为第i个块压缩的key

第二分支：细粒度选择



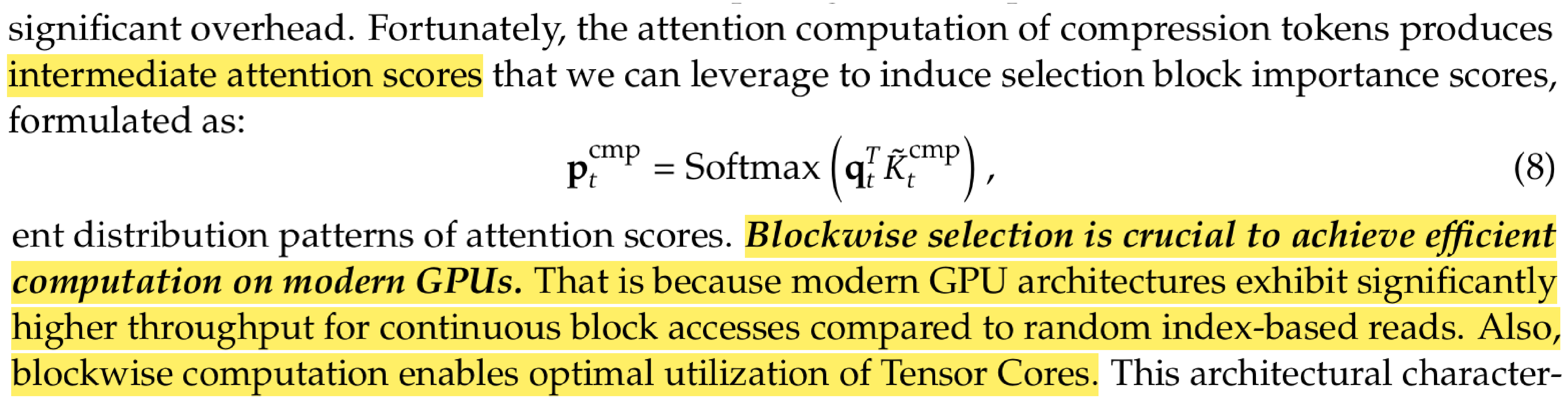
Using only compressed keys, values might lose important fine-grained information, motivating us to selectively preserve individual keys, values. Below we describe our efficient token selection mechanism that identifies and preserves the most relevant tokens with low computational overhead.



Prior works (Jiang et al., 2024) have shown that attention scores often exhibit spatial continuity, suggesting that neighboring keys tend to share similar importance levels. Our visualization in

- 用第一层的分数，判断哪一粗块值得细挖
- 找到高分粗块对应的原始 token 范围
- 细块是大于或等于粗块的
- 用高分的细块里的token来计算注意力

第二分支：细粒度选择



significant overhead. Fortunately, the attention computation of compression tokens produces intermediate attention scores that we can leverage to induce selection block importance scores, formulated as:

$$\mathbf{p}_t^{\text{cmp}} = \text{Softmax}\left(\mathbf{q}_t^T \tilde{K}_t^{\text{cmp}}\right), \tag{8}$$

ent distribution patterns of attention scores. ***Blockwise selection is crucial to achieve efficient computation on modern GPUs.*** That is because modern GPU architectures exhibit significantly higher throughput for continuous block accesses compared to random index-based reads. Also, blockwise computation enables optimal utilization of Tensor Cores. This architectural character-

- 粗粒度压缩阶段的注意力计算已产生中间注意力分数

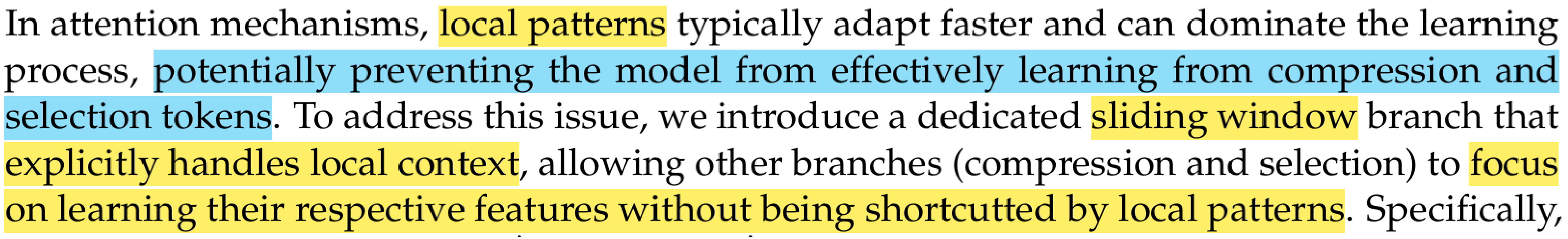
  可以复用

- 适配GPU

  跳着读不如整块读

  GPU 对连续块内存访问的吞吐量远高于随机单个 token 访问

第三分支：滑动窗口


In attention mechanisms, local patterns typically adapt faster and can dominate the learning process, potentially preventing the model from effectively learning from compression and selection tokens. To address this issue, we introduce a dedicated sliding window branch that explicitly handles local context, allowing other branches (compression and selection) to focus on learning their respective features without being shortcutted by local patterns. Specifically,

- 只保留最近的w个token

  文中取的是512
- 给滑动窗口层独立的 KV 缓存

  防止交叉影响

门控机制

$$\mathbf{o}_t^* = \sum_{c \in C} g_t^c \cdot \mathrm{Attn}(\mathbf{q}_t, \tilde{K}_t^c, \tilde{V}_t^c).$$

- 用上述公式合并三个分支的结果

  三个分支的计算结果求加权平均

  g_t^c即为门控分数
- g_t^c如何计算?

  MLP和sigmoid

组件均可学习

- 过往的稀疏注意力存在的问题
  - Openai的稀疏注意力
    - 无任何可学习组件
  - NSA
    - 压缩层的φ是可学的
    - 选择层的分数是可学的
    - 门控层的权重是可学的
    - 支持梯度回传来不断调整参数