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

Jingyang Yuan[*1,2], Huazuo Gao[1], Damai Dai[1], Junyu Luo[2], Liang Zhao[1], Zhengyan Zhang[1], Zhenda Xie[1], Y. X. Wei[1], Lean Wang[1], Zhiping Xiao[3], Yuqing Wang[1], Chong Ruan[1], Ming Zhang[2], Wenfeng Liang[1], Wangding Zeng[1]

[1]DeepSeek-AI
[2]Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University, PKU-Anker LLM Lab
[3]University of Washington
{yuanjy, mzhang_cs}@pku.edu.cn, {zengwangding, wenfeng.liang}@deepseek.com

arXiv preprint 2025

HUMANE Lab 김태균

2025.03.07

Background

- The existing attention mechanisms poses challenges in long-context modeling due to high computational costs
- To address this, sparse attention is used
- However, limitations still remain

Limitations of existing sparse attention

1. Unable to fully exploit sparse attention
2. Inference-only approaches

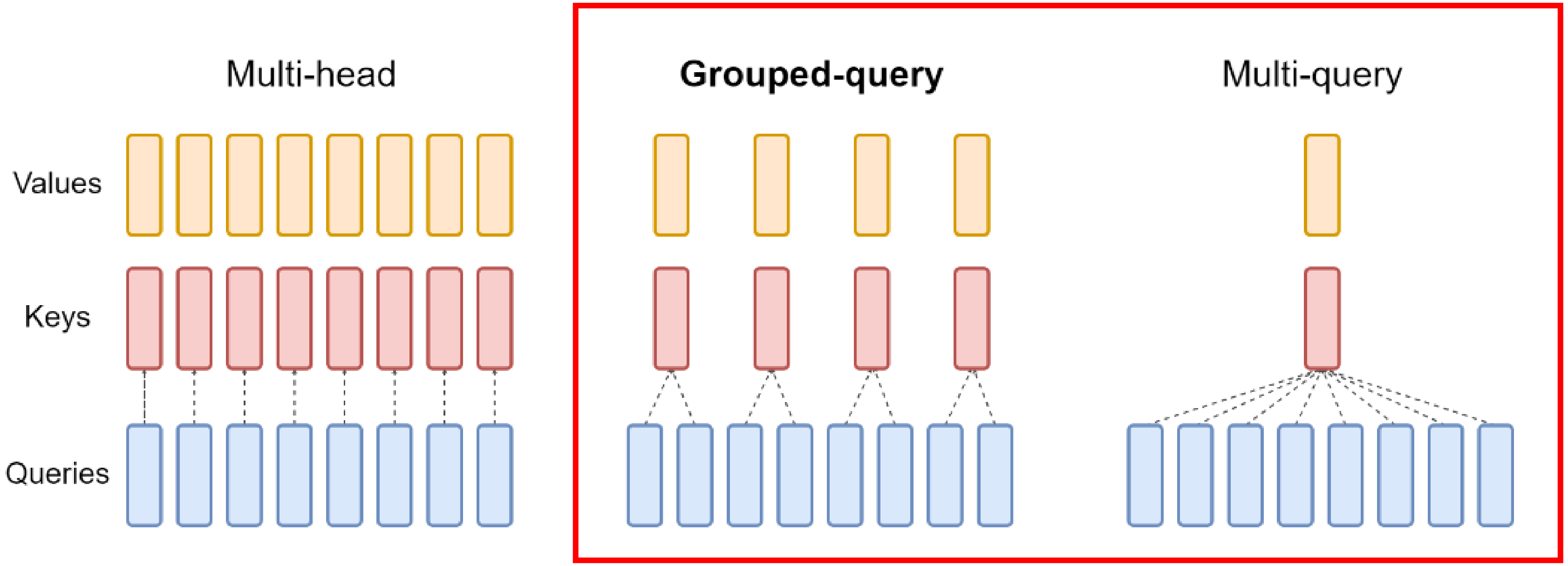
1. Unable to fully exploit sparse attention

- Phase-restricted sparsity
  - prefilling (e.g. MInference)
  - autoregressive decoding (e.g. H2O)

=> The remaining phases still incur computational costs

1. Unable to fully exploit sparse attention

- Incompatibility with advanced attention architecture



2. Inference-only approaches

- Performance degradation
  - Applying sparsity post-hoc forces models to deviate from their pretrained optimization trajectory

=> Efforts to adapt sparse attention for training are demanding

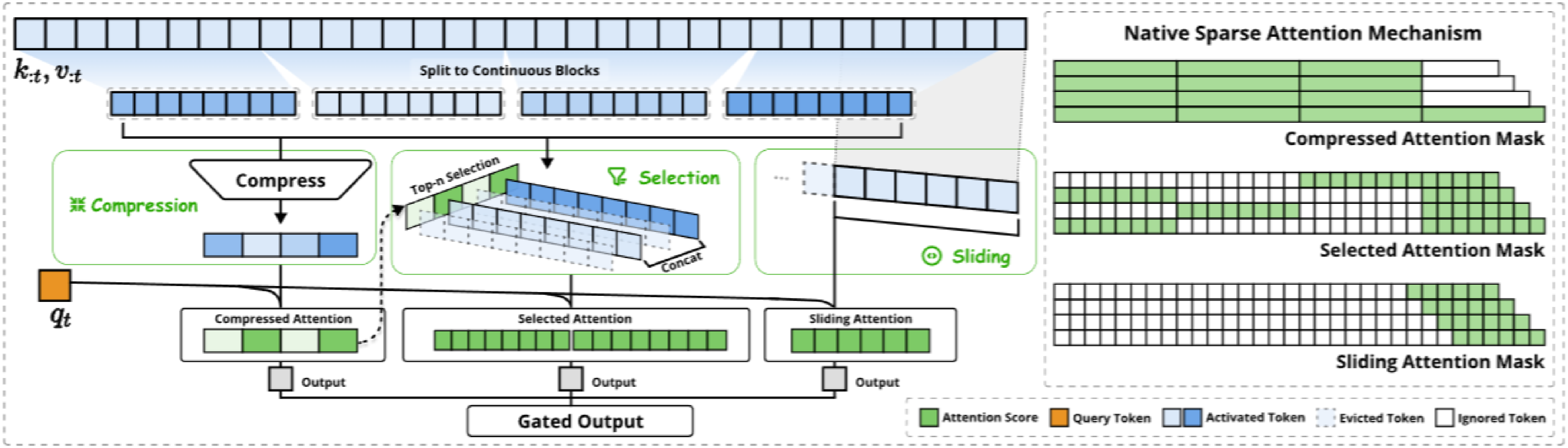
Native Sparsity as an Imperative

- Therefore, sparse attention that can be applied in both computational efficiency and the training process is needed

Native Sparse Attention (NSA)

: Hardware-aligned and natively trainable sparse attention

Overview

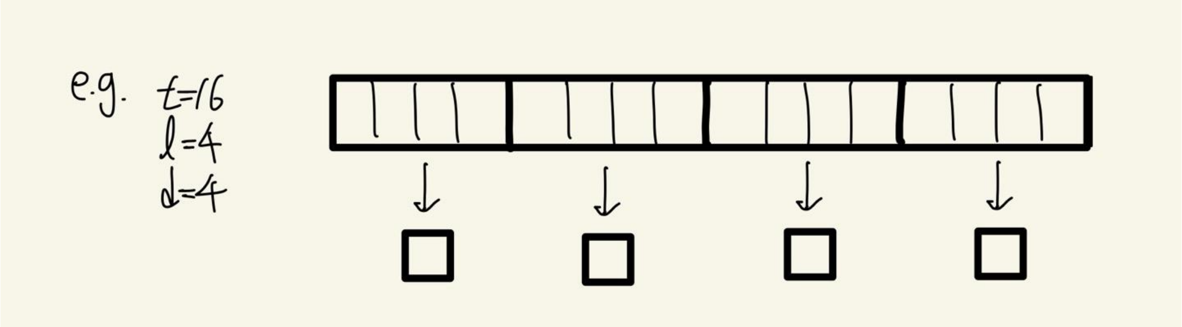


$$\mathbf{o}_t^* = \sum_{c \in C} g_t^c \cdot \mathrm{Attn}(\mathbf{q}_t, \tilde{K}_t^c, \tilde{V}_t^c)$$

1. Token Compression

- Aggregating sequential blocks of keys or values into block-level representations
  - φ : MLP map keys in a block to a single compressed key
  - l : block length
  - d : sliding stride

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

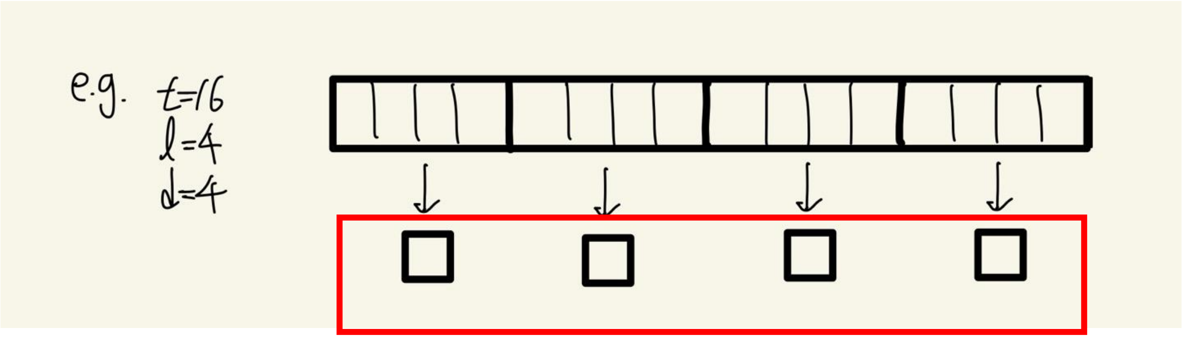


2. Token Selection

- Divide key, value sequences into selection blocks
- Then assign importance scores to each block
    - case 1 : l = l’
        - l’ : selection block size

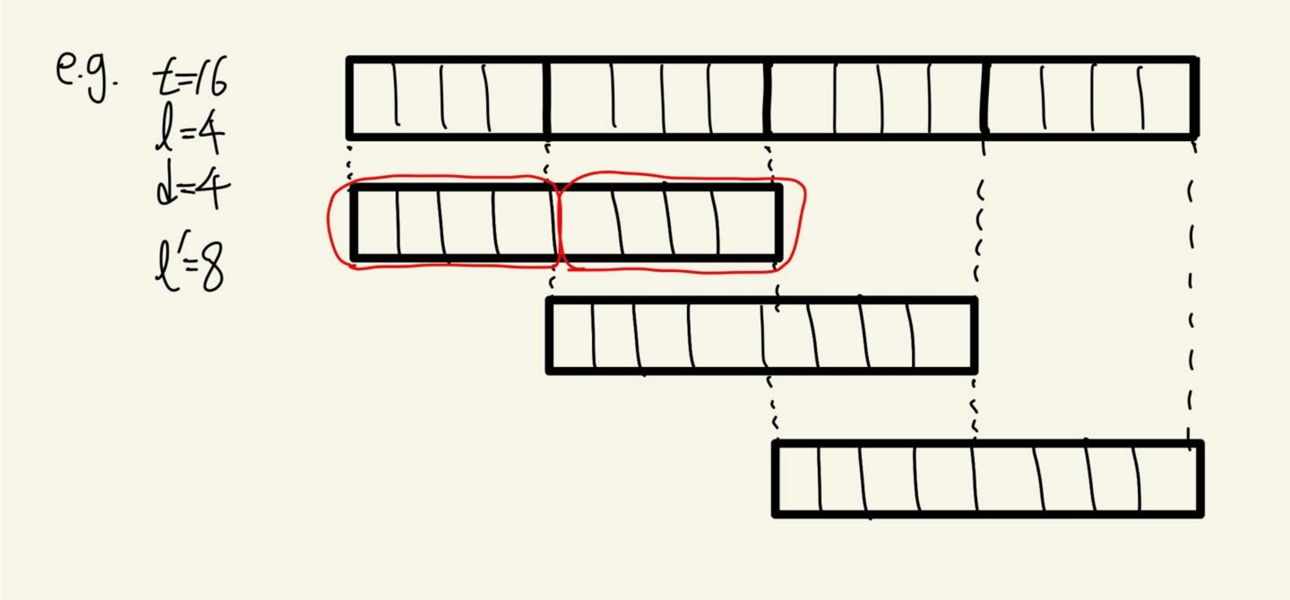
$$\mathbf{p}_t^{\text{cmp}} = \text{Softmax}\left(\mathbf{q}_t^T \tilde{K}_t^{\text{cmp}}\right)$$

$$\mathbf{p}_t^{\text{slc}} = \mathbf{p}_t^{\text{cmp}}$$



2. Token Selection

- Divide key, value sequences into selection blocks
- Then assign importance scores to each block
  - case 2 : l ≠ l'
    - l ≤ l', d|l, d|l'



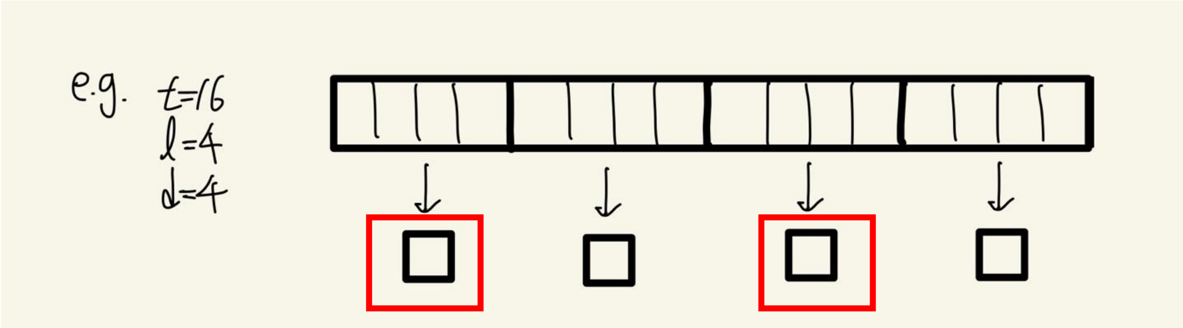
$$\mathbf{p}_t^{\mathrm{cmp}} = \mathrm{Softmax}\left(\mathbf{q}_t^T \tilde{K}_t^{\mathrm{cmp}}\right)$$

$$\mathbf{p}_t^{\mathrm{slc}}[j] = \sum_{m=0}^{\frac{l'}{d}-1} \sum_{n=0}^{\frac{l}{d}-1} \mathbf{p}_t^{\mathrm{cmp}}\left[\frac{l'}{d} j - m - n\right]$$

$$\mathbf{p}_t^{\mathrm{slc}'} = \sum_{h=1}^{H} \mathbf{p}_t^{\mathrm{slc},(h)}$$

2. Token Selection

- Divide key, value sequences into selection blocks
- Then assign importance scores to each block
- Retain tokens within the top-n sparse blocks ranked by block importance scores

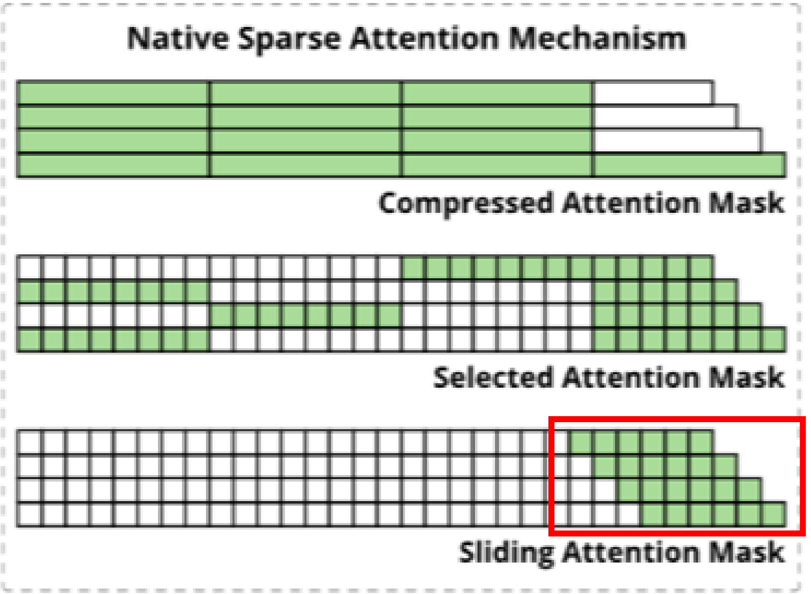


$$\mathcal{I}_t = \{i \mid \mathrm{rank}(\mathbf{p}_t^{\mathrm{slc}\prime}[i]) \leqslant n\}$$

$$\tilde{K}_t^{\mathrm{slc}} = \mathrm{Cat}\left[\{\mathbf{k}_{il'+1:(i+1)l'} \mid i \in \mathcal{I}_t\}\right]$$

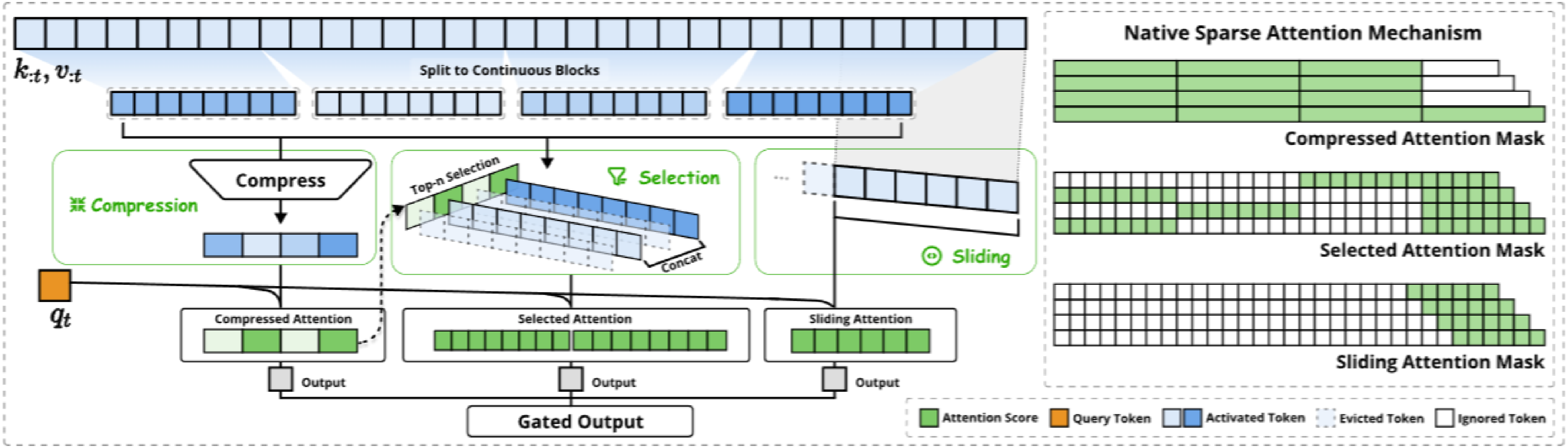
3. Sliding Window

- Maintain recent tokens in a window to handles local context



$$\tilde{K}_t^{\text{win}} = \mathbf{k}_{t-w:t}, \tilde{V}_t^{\text{win}} = \mathbf{v}_{t-w:t}$$

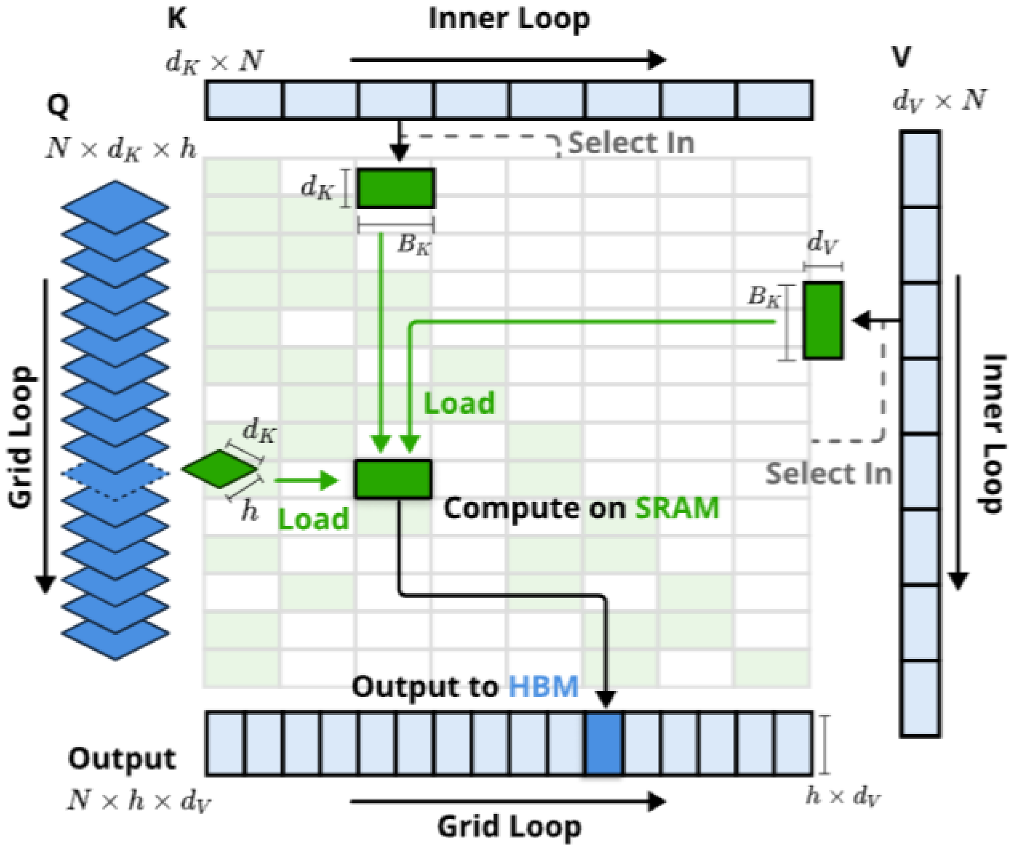
Overview



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

4. Hardware-aligned sparse attention kernel

- Group-centric data loading
- Shared KV fetching
- Outer loop on grid



Experiments

1. General benchmarks evaluation
2. Long-context benchmarks evaluation
3. Reasoning performance

Experiments

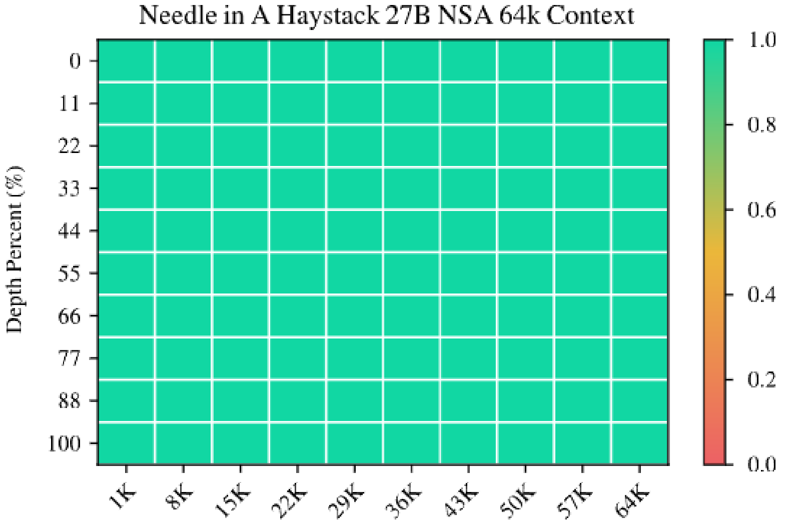
1. General benchmarks evaluation

=> NSA outperforms in most benchmarks

Model	MMLU Acc. 5-shot	MMLU-PRO Acc. 5-shot	CMMLU Acc. 5-shot	BBH Acc. 3-shot	GSM8K Acc. 8-shot	MATH Acc. 4-shot	DROP F1 1-shot	MBPP Pass@1 3-shot	HumanEval Pass@1 0-shot	Avg.
Full Attn	**0.567**	0.279	0.576	0.497	0.486	0.263	0.503	**0.482**	0.335	0.443
NSA	0.565	**0.286**	**0.587**	**0.521**	**0.520**	**0.264**	**0.545**	0.466	**0.348**	**0.456**

Experiments

2. Long-context benchmarks evaluation



Model	SQA			MQA				Synthetic		Code	Avg.
	MFQA-en	MFQA-zh	Qasper	HPQ	2Wiki	GovRpt	Dur	PassR-en	PassR-zh	LCC	
H2O	0.428	0.429	0.308	0.112	0.101	0.231	0.208	0.704	0.421	0.092	0.303
InfLLM	0.474	0.517	0.356	0.306	0.250	0.277	0.257	0.766	0.486	0.143	0.383
Quest	0.495	0.561	0.365	0.295	0.245	0.293	0.257	0.792	0.478	0.135	0.392
Exact-Top	0.502	0.605	0.397	0.321	0.288	0.316	0.291	0.810	0.548	0.156	0.423
Full Attn	**0.512**	0.623	0.409	0.350	0.305	**0.324**	0.294	0.830	**0.560**	0.163	0.437
NSA	0.503	**0.624**	**0.432**	**0.437**	**0.356**	0.307	**0.341**	**0.905**	0.550	**0.232**	**0.469**

Figure 5 | Needle-in-a-Haystack retrieval accuracy across context positions with 64k context length. NSA achieves perfect accuracy through its hierarchical sparse attention design.

Experiments

3. Reasoning performance

Generation Token Limit	8192	16384
Full Attention-R	0.046	0.092
NSA-R	**0.121**	**0.146**

Table 3 | AIME Instruction-based Evaluating after supervised fine-tuning. Our NSA-R demonstrates better performance than Full Attention-R at both 8k and 16k sequence lengths

Conclusion

- NSA : A hardware-aligned sparse attention architecture for efficient long-context modeling
- Integrating hierarchical token compression with blockwise token selection within a trainable architecture
- Achieves accelerated training and inference while maintaining Full Attention performance

Open Question

- Can NSA be applied to multimodal models?