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

Amit Kumar

AI/NLP Engineer at E42.ai

Table of contents

1. Introduction

2. Native Sparse Attention (NSA)

3. Token Compression Attention

4. Token selection attention

5. Sliding window attention

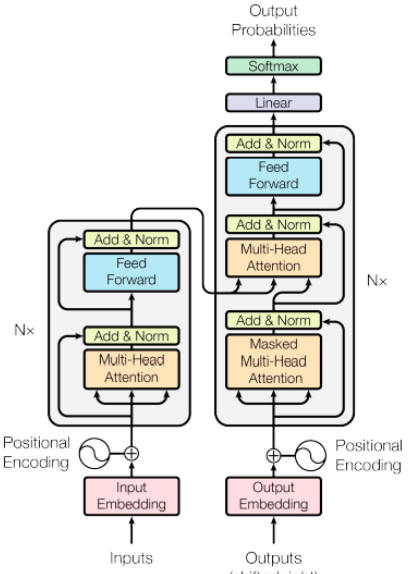
6. Results

7. Conclusion

Introduction

- Long-context modeling is critical for next-gen language models.
- Standard attention has quadratic complexity: $O(N^2)$ for sequence length N.
- This leads to huge computational and memory costs for long sequences (e.g., 64K tokens).
- Existing attention methods:
  1. Often hardware-inefficient
  2. Lack end-to-end trainability
  3. Use fixed heuristics limiting adaptability

Attention Definition and Intuition



$$O = f(QK^T)V \qquad \begin{aligned} Q &\in \mathbb{R}^{N,d} \\ K &\in \mathbb{R}^{M,d} \\ V &\in \mathbb{R}^{M,D} \\ O &\in \mathbb{R}^{N,D} \end{aligned}$$

Attention formulation

$$O = f(QK^T)V \qquad \begin{aligned} Q &\in \mathbb{R}^{B,H,N,d} \\ K &\in \mathbb{R}^{B,H,M,d} \\ V &\in \mathbb{R}^{B,H,M,D} \\ O &\in \mathbb{R}^{B,H,N,D} \end{aligned}$$

Attention formulation with batch and head dims

Attention Equation Visualization

(N, d) (d, M) (M, d)

$$O = Softmax(QK^T)V$$

Literally just a lucky choice of function that is fantastic

(d)

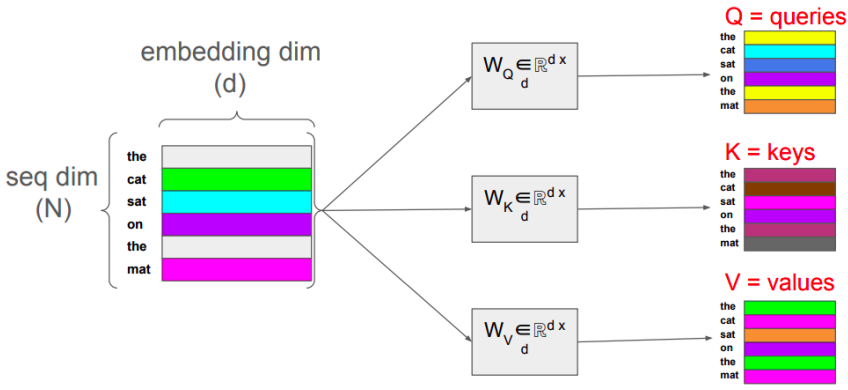
(d) (d) (d)

$$O_t = \frac{\sum_{s=1}^{M} e^{Q_t K_s^T} V_s}{\sum_{s=1}^{M} e^{Q_t K_s^T}}$$

(d) (d)

(1) scalar

$$Q \in \mathbb{R}^{N,d}$$
$$K \in \mathbb{R}^{M,d}$$
$$V \in \mathbb{R}^{M,D}$$
$$O \in \mathbb{R}^{N,D}$$


embedding dim
(d)
seq dim
(N)
the
cat
sat
on
the
mat
$W_Q \in \mathbb{R}^{d \times d}$
$W_K \in \mathbb{R}^{d \times d}$
$W_V \in \mathbb{R}^{d \times d}$
Q = queries
K = keys
V = values
still of shape (N, d). However, we will be playing around with the shapes later.
The colors really mean nothing at this point lol

Q = queries

K = keys

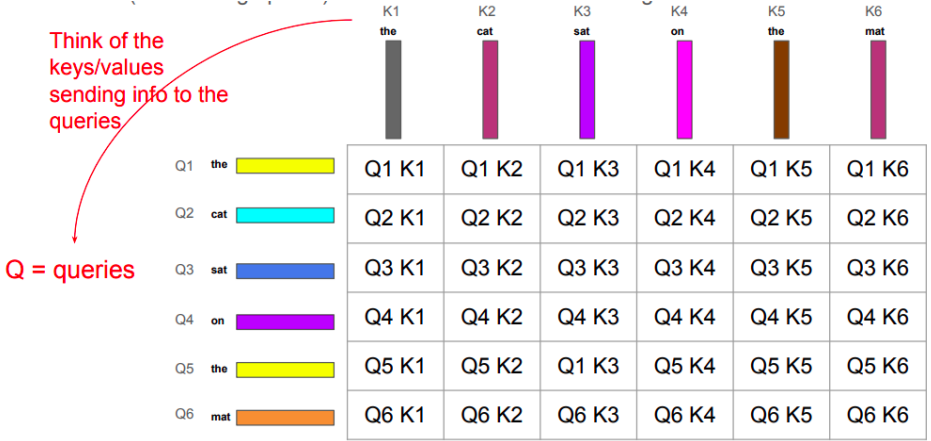
V = values

$$O_{\text{sat}} = \frac{e^{Q_{\text{sat}}K^T_{\text{the}}}V_{\text{the}} + e^{Q_{\text{sat}}K^T_{\text{cat}}}V_{\text{cat}} + e^{Q_{\text{sat}}K^T_{\text{sat}}}V_{\text{sat}} + e^{Q_{\text{sat}}K^T_{\text{on}}}V_{\text{on}} + e^{Q_{\text{sat}}K^T_{\text{the}}}V_{\text{the}} + e^{Q_{\text{sat}}K^T_{\text{mat}}}V_{\text{mat}}}{e^{Q_{\text{sat}}K^T_{\text{the}}} + e^{Q_{\text{sat}}K^T_{\text{cat}}} + e^{Q_{\text{sat}}K^T_{\text{sat}}} + e^{Q_{\text{sat}}K^T_{\text{on}}} + e^{Q_{\text{sat}}K^T_{\text{the}}} + e^{Q_{\text{sat}}K^T_{\text{mat}}}}$$

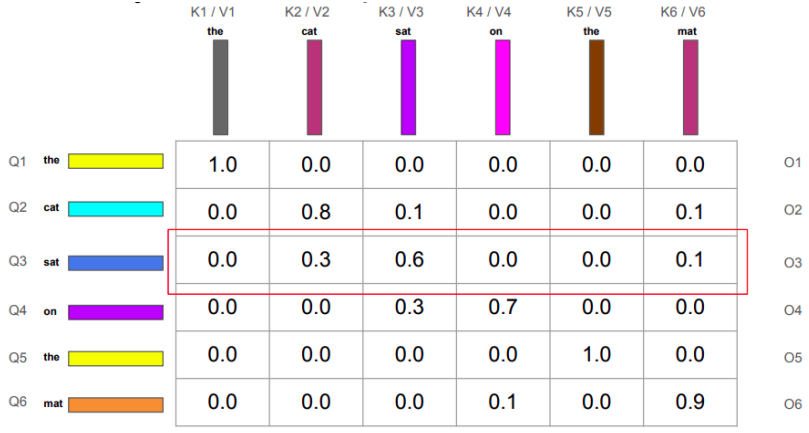
Notice how the query determines the word we are modeling in the output

We do this for all words in the sequence!

Attention matrix



What exactly does this represent?

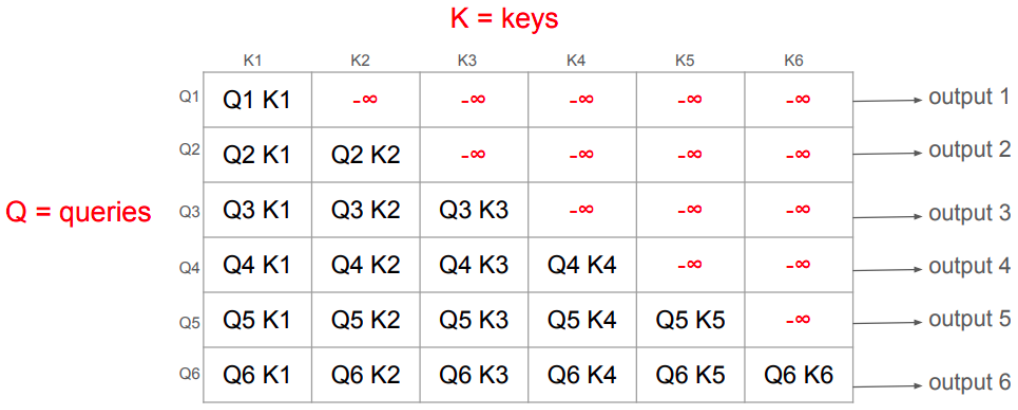


$$O_3 = e^{Q_3K_1}V_1 + e^{Q_3K_2}V_2 + e^{Q_3K_3}V_3 + e^{Q_3K_4}V_4 + e^{Q_3K_5}V_5 + e^{Q_3K_6}V_6$$

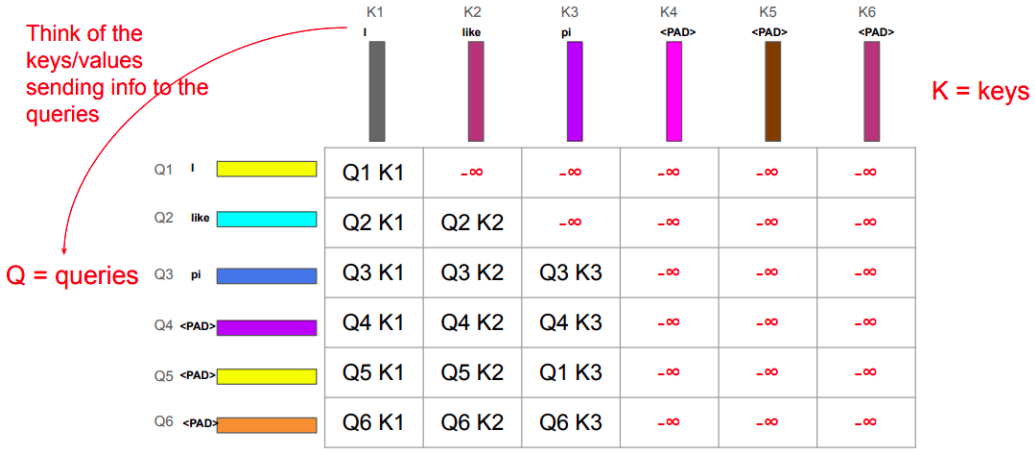
$$O_3 = 0.0 * V_1 + 0.3 * V_2 + 0.6 * V_3 + 0.0 * V_4 + 0.0 * V_5 + 0.1 * V_6$$

$$O_{\text{sat}} = 0.0 * V_{\text{the}} + 0.3 * V_{\text{cat}} + 0.6 * V_{\text{sat}} + 0.0 * V_{\text{on}} + 0.0 * V_{\text{the}} + 0.1 * V_{\text{mat}}$$

Causal attention



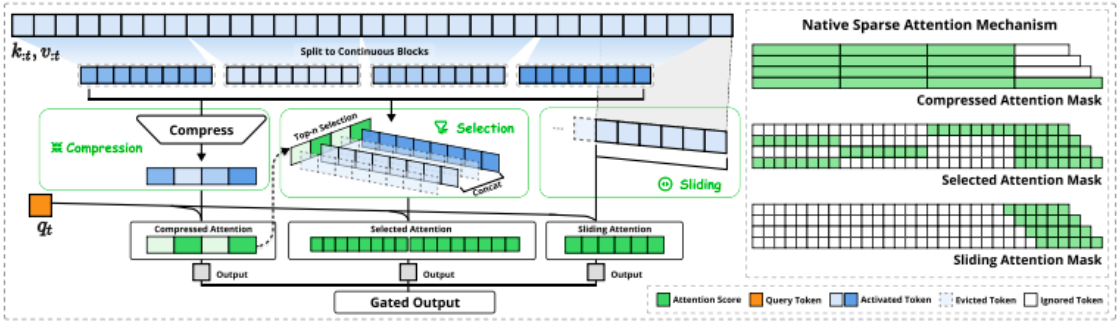
Causal Padding Mask



What is Native Sparse Attention (NSA)?

- A natively trainable sparse attention mechanism designed for:
  - Hardware alignment (efficient on GPUs/TPUs)
  - End-to-end differentiability (learns sparse patterns during training)
  - Efficient long-context modeling
- Combines algorithmic innovations with hardware-aware optimizations
- Input sequence split into blocks (e.g., 32 tokens/block)
- Three complementary pathways:
  - Compression: Learnable MLP compresses blocks into summary vectors
  - Selection: Differentiable top-k selects important blocks globally
  - Sliding Window: Local attention on recent tokens (e.g., 512 tokens)
- Balances global context awareness and local precision

NSA architecture



Compression Pathway in Native Sparse Attention

- The input sequence of tokens is partitioned into **blocks** of size B.
- Each block of tokens is passed through a **learnable MLP** (Multi-Layer Perceptron) with positional encoding.
- The MLP compresses the block into a **compact summary vector** representing the block.
- This reduces the global attention complexity from $O(N^2)$ to approximately $O(N \times \frac{N}{B})$, which is near-linear.

Simple example:

- Suppose sequence length $N = 16$, block size $B = 4$.
- The sequence is divided into $M = \frac{N}{B} = 4$ blocks.
- Each block of 4 tokens is compressed into 1 summary vector.
- Global attention attends to these 4 summary vectors instead of all 16 tokens.

Block-Level Compressed Key Representation (Simplified)

Idea: Instead of using every single key or value, we group them into blocks. Each block is turned into a single, compressed key (or value) that summarizes the whole block. This makes processing faster and helps capture the main information.

- Take blocks of length l from the sequence of keys.
- Move forward by d steps to get the next block (stride).
- Use a function φ (MLP) to turn each block into one compressed key.

Equation:

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

Notes:

- l: block size
- d: step size between blocks
- φ: function that compresses a block into a single key

Leveraging Attention Scores for Block Importance

- Computing block importance scores can introduce significant overhead.
- Attention computation of compression tokens produces intermediate attention scores.
- These scores can be leveraged to induce selection block importance scores:

$$p_{cmp}^{t} = \text{Softmax}\left(q_t^T \widetilde{K}_{cmp}^{t}\right),$$

where $p_{cmp}^{t} \in \mathbb{R}^{\lfloor \frac{t-l}{d} \rfloor + 1}$ represents attention scores between q_t and compression keys $\widetilde{K}_{cmp}^{t}$.

Top-n Block Selection

- Retain tokens within top-n sparse blocks ranked by importance scores.

$$I_t = \{i \mid \text{rank}(p_{slc}^{t'}[i]) \leq n\}$$

$$\widetilde{K}_{slc}^{t} = \text{Cat}\{k_{il'+1:(i+1)l'} \mid i \in I_t\}$$

- $\text{rank}(\cdot)$ ranks scores in descending order (rank 1 = highest).
- $\widetilde{K}_{slc}^{t} \in \mathbb{R}^{d_k \times nl'}$ is the tensor of selected compression keys.
- Selected keys and values then participate in attention computation with q_t.

Sliding Window attention

- Local patterns adapt faster and can dominate learning, hindering compression and selection token learning.
- Introduce a *sliding window* dedicated to local context, allowing other branches to focus on their features without shortcutting by local patterns.
- **Mechanism:**
  - Maintain recent tokens in window w:

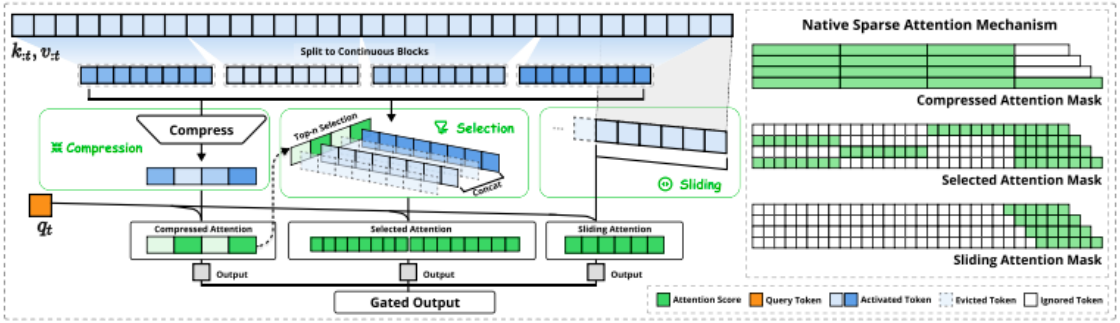
$$\tilde{K}_t^{win} = k_{t-w:t}, \quad \tilde{V}_t^{win} = v_{t-w:t}$$

  - Separate attention computations into three branches:

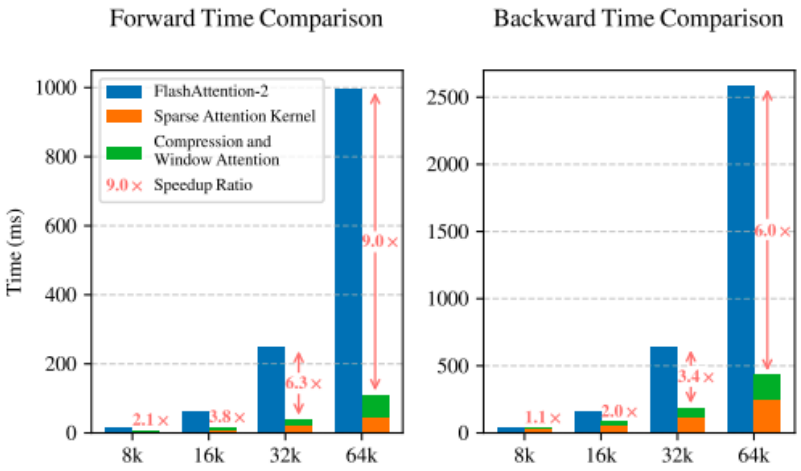
$$\text{Compression: } (\tilde{K}_t^{cmp}, \tilde{V}_t^{cmp}), \quad \text{Selection: } (\tilde{K}_t^{slc}, \tilde{V}_t^{slc}), \quad \text{Sliding Window: } (\tilde{K}_t^{win}, \tilde{V}_t^{win})$$

  - Use independent keys and values per branch to prevent shortcut learning and gradient interference.
  - Aggregate branch outputs via a learned gating mechanism.

NSA architecture



Comparison of NSA kernel with FlashAttention-2



Context Length	8192	16384	32768	65536
Full Attention	8192	16384	32768	65536
NSA	2048	2560	3584	5632
Expected Speedup	**4**×	**6.4**×	**9.1**×	**11.6**×

NSA achieving lower loss values

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**

Conclusion

- **Efficient Long-Context Modeling**: NSA uses hierarchical token compression and blockwise token selection to handle very long input sequences efficiently.
- **Hardware-Optimized Design**: It is specially designed to work well with modern GPUs, reducing memory access and speeding up training and inference.
- **Maintains High Accuracy**: NSA matches or exceeds the performance of full attention models on benchmarks and long-context tasks.
- **Significant Speedup**: It achieves much faster computation (up to 9× faster in some cases) while keeping the model's reasoning and accuracy intact.

Thank You