The Sparse Frontier: Sparse Attention Trade-offs in Transformer LLMs

Piotr Nawrot   Robert Li   Renjie Huang
Sebastian Ruder   Kelly Marchisio   Edoardo M. Ponti

University of Edinburgh, Meta, Cohere

The Need for Long-Sequence Processing

Modern AI applications require processing vast amounts of text:

- **Financial Analysis**: Processing entire 10-K reports (100K+ tokens)
- **Literature**: Analyzing full novels like "War and Peace" (600K+ tokens)
- **Code Understanding**: Processing entire codebases (millions of tokens)

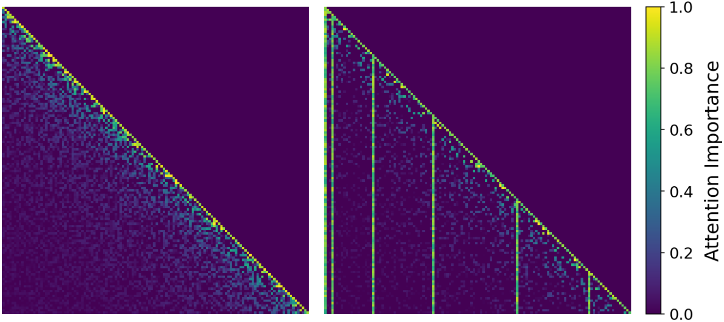
Current AI models struggle with these long inputs due to computational limits.

The Self-Attention Bottleneck

Challenges of Long-Sequence Processing in LLMs:

- *Prefilling*: Quadratic complexity of dense attention ($O(n^2)$).
- *Decoding*: Linearly growing KV cache, high memory bandwidth usage.

Sparse Attention as a Solution



- **Assumption:** We don't need to compute all $O(N^2)$ interactions between elements in the sequence.
- **Benefit:** Faster model as we skip computations.
- **Challenge:** How to know which interactions to compute?

Motivation for This Work

arXiv Advanced Search (Last 12 Months)

- **Search Terms:** "sparse attention" OR "efficient attention" OR "kv cache"
- **Results:** 418 papers

Yet, comprehensive evaluation is missing:

- What are the key differences between different sparse attention methods, and which method is the best?
- What is the maximum sparsity level that maintains dense performance across diverse tasks?
- What is better: large sparse model or small dense one?

Our Approach 1 / 2

We surveyed sparse attention methods:

- Four key design axes: units of sparsification, importance estimation, budget allocation, KV cache management.
- Six representative patterns with unified implementations allowing for systematic ablations.

We surveyed long context evaluation:

- To stress test sparse attention methods we designed an evaluation suite that covers diverse task types, and both natural and synthetic inputs.

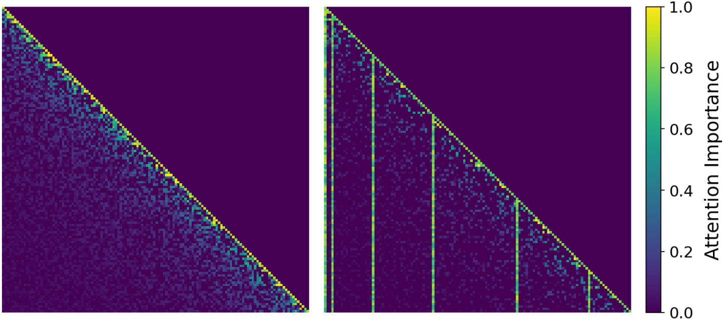
Our Approach 2 / 2

Finally, we ran a lot of experiments...

- We tested sequence lengths from 16k to 128k tokens.
- We tested model sizes from 7B to 72B parameters.
- We tested sparsity levels from 0% to 95% ($20\times$).

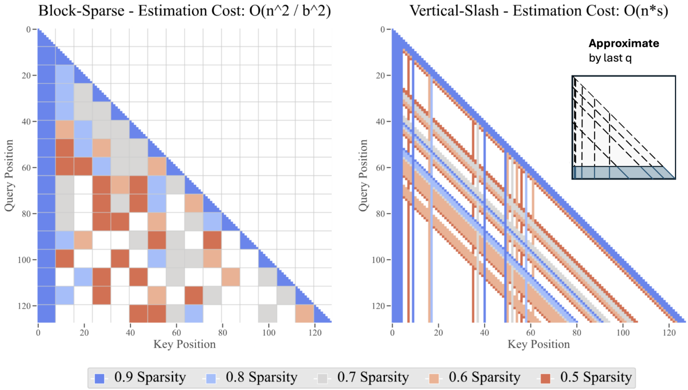
Making it the largest-scale empirical analysis to date of training-free sparse attention methods.

Main Challenge of Sparse Attention



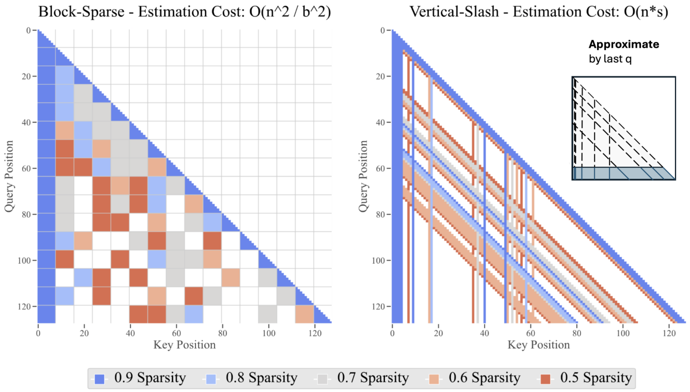
- Attention maps are sparse, but also irregular. **They differ across tasks, models, layers, and heads.**
- How do we know which subset of interactions to compute?

Two SOTA Methods - Units of Sparsification



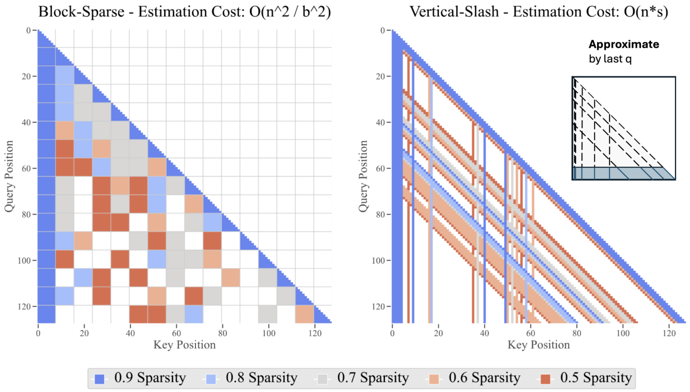
- **Block-Sparse**: Blocks of size B x B (e.g. 64x64).
- **Vertical-Slash**: Columns and Diagonals.

Two SOTA Methods - Importance Estimation



- **Block-Sparse**: Estimates importance of each block using heuristic block representations.
- **Vertical-Slash**: Estimates importance of each column and diagonal using suffix tokens.

Two SOTA Methods - Budget Allocation

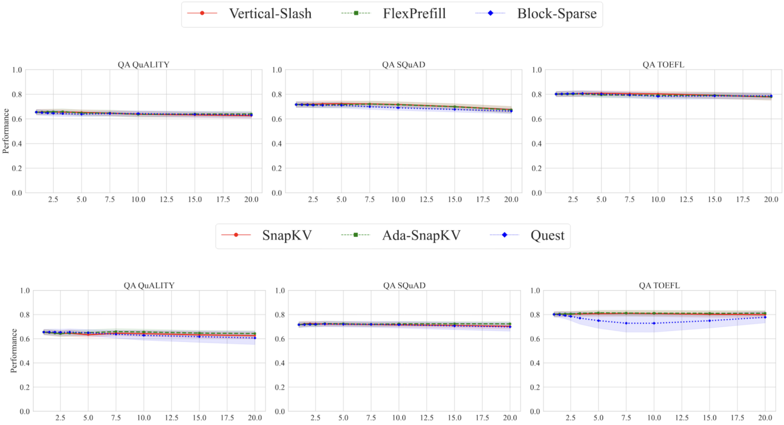


- **Block-Sparse**: Chooses top-k blocks in each row, for each layer and head.
- **Vertical-Slash**: Chooses top-k columns and diagonals, for each layer and head.

Results

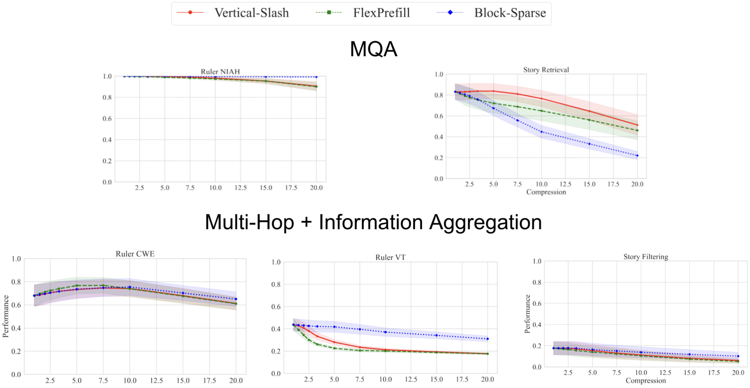
1. No universal method exists yet, there are 1) task-dependent and 2) design-related trade-offs.
2. High sparsity ($\geq 90\%$) is tolerated on average, although high variance is observed.
3. Large sparse models are better than small dense models.

Results #1.1: Task Trade-offs (QA)



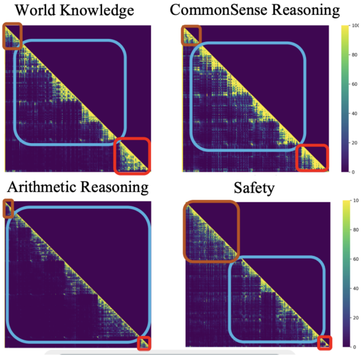
- Single Question Answering tasks tolerate very high sparsity levels during both prefilling and decoding.

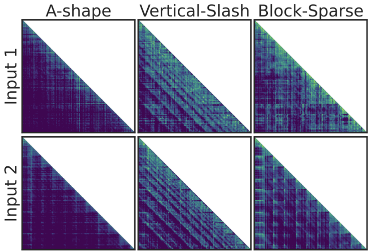
Results #1.1: Task Trade-offs (MQA vs Reasoning)



- For more complex tasks, we need to attend to more tokens, which results in smaller sparsity.
- Different methods excel for different tasks.

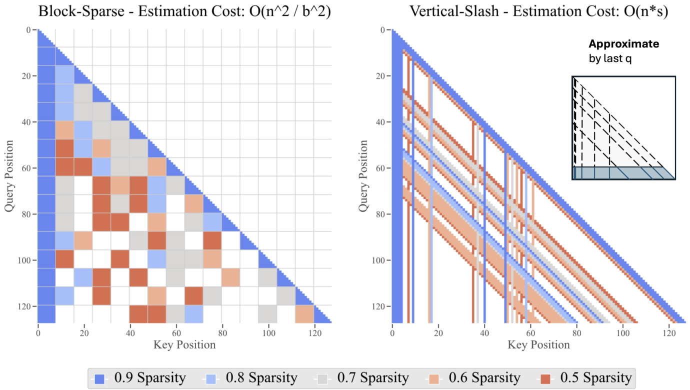
Results #1.1: Task Trade-offs (MQA vs Reasoning)





- Again, it's because - for more difficult tasks, Attention Maps vary across inputs, models, layers, and heads.

Results #1.2: Design Trade-offs (Flexibility)



- Why can't we be more flexible and do better? What are the trade-offs?
- Why can't we compute top-k token-to-token interactions?

Results #1.2: Design Trade-offs (Flexibility)

- Cost of Dense Attention is $O(n^2)$.
- Cost of Sparse Attention is cost of sparse computation + cost of **index building**.
- Let's assume that for each query (N) we compute a fixed number of interactions (K). Then cost of sparse computation is $O(NK)$.
- We need to estimate what to compute in time $\leq O(N^2)$.
  - **Estimating importance of each query-key interaction is infeasible as it requires $O(n^2)$ computation.**

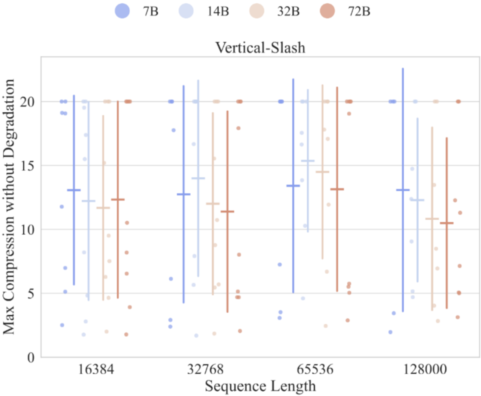
Results #1.2: Design Trade-offs (Flexibility)

	Individual Keys	**Chunks of Keys**
Individual Queries	No	Maybe
Chunks of Queries	Maybe	Yes
All Queries	Yes	Yes

Feasible approaches must estimate importance efficiently:

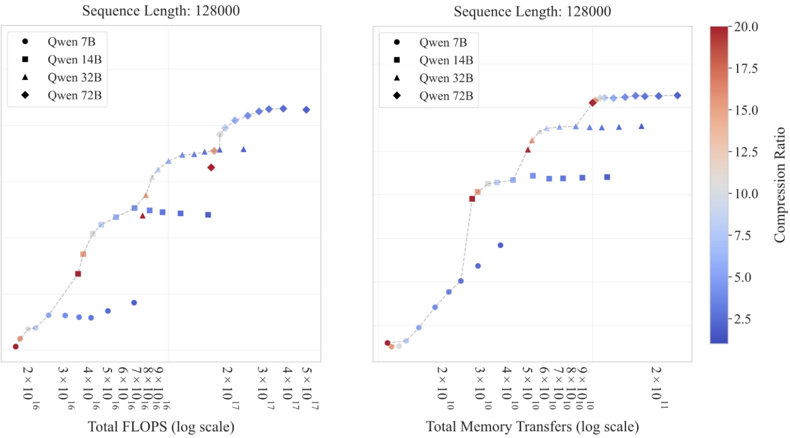
- Individual-Individual: Requires $O(n^2)$ estimation cost
- Individual-Chunk & Chunk-Individual: Moderate estimation overhead
- Chunk-Chunk & All-Chunk: Efficient estimation possible

Results #2: Maximum Sparsity vs Model Size



- Average attention compression which retains dense performance exceeds 12×.
- There is no clear impact of model size and sequence length.
- Variance is very high, hence careful deployment is required.

Results #3: Large Sparse Models Are Better



- Sparse configurations dominate Cost-Efficiency Pareto Frontier. It's more cost-effective to use sparse models.

Conclusions

Which method is the best?

- No universal method exists yet. Our work illustrates trade-offs between different methods guiding future research.

How much sparsity we can use to maintain dense performance?

- Models tolerate high average sparsity ($15\times$). Moderate compression can degrade performance on certain tasks, mandating careful pre-deployment evaluation.

What is better: large sparse model or small dense one?

- Sparse models are more effective for the same computational cost. Most of the recent frontier LLMs have some form of sparse attention. We expect this trend to continue.

Open Source Repositories

Two complementary repositories for sparse attention research:

- **sparse-frontier**
  `github.com/PiotrNawrot/sparse-frontier`
  - Production-ready evaluation framework with vLLM integration
  - 6 SOTA sparse attention implementations
  - Comprehensive evaluation suite across 9 diverse tasks
  - Supports 100+ models from 7B to 405B parameters
- **nano-sparse-attention**
  `github.com/PiotrNawrot/nano-sparse-attention`
  - Educational PyTorch implementations for learning
  - Perfect starting point before diving into optimized code