MiniMax Sparse Attention: A Small MacBook-Scale Research Note

Vuk Rosic

June 14, 2026

Abstract

This note defines an open-source research program around MiniMax-style sparse attention. The immediate goal is not to claim a new state-of-the-art model. The goal is to make a small, inspectable implementation whose routing behavior can be checked, ablated, and compared against dense attention as context length grows. The current evidence shows that dense and sparse variants remain close on ordinary next-token loss at small scale, while top-k and block size visibly change routing density, throughput, and loss. That makes the project useful, but unfinished: the central work now is correctness verification, dense-oracle diagnostics, long-context retrieval stress tests, and architecture ablations that explain when sparse routing helps or fails.

1 Goal

The research question is:

> Can a block-indexed sparse attention mechanism keep the important context tokens while reducing long-context attention cost, and can we prove that behavior with small reproducible experiments?

This is different from simply training a small language model and reporting validation loss. Sparse attention can look fine on short next-token prediction while failing at the job it is supposed to do: preserve useful information from long prefixes. For that reason, the project has four proof obligations.

1. **Implementation correctness.** The causal mask, block selection, grouped-query mapping, and sparse gather path must match the intended mechanism.
2. **Routing fidelity.** The selected blocks should overlap with blocks that dense attention would actually use.
3. **Long-context usefulness.** The sparse model should survive longer context windows and retrieval-style probes, not only short training contexts.
4. **Ablation clarity.** Each architectural change should answer one question about routing, fallback, reuse, or capacity.

2 Reference Mechanism

Figure 1 is the source diagram used as the implementation reference. The repository does not treat this diagram as a fixed endpoint; it treats it as the first point in a larger sparse-attention design space. The implementation should therefore be easy to inspect, test, and modify.

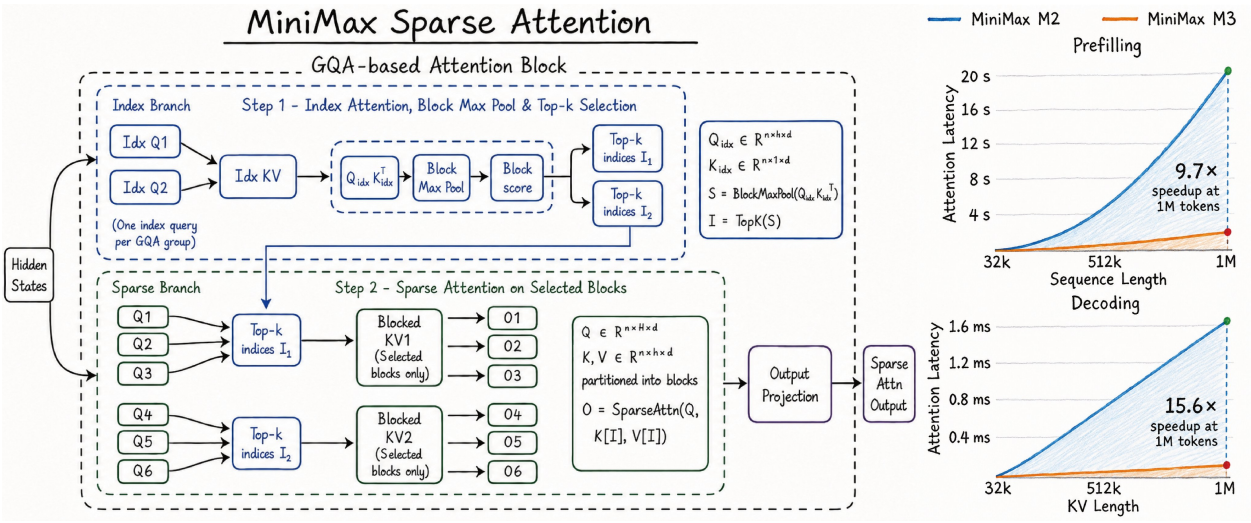

Figure 1: Reference MiniMax-style sparse attention diagram. Source: Skyler Miao X post, `https://x.com/SkylerMiao7/status/2059285750458544561`.

3 Research Framing

This paper should be read as an experiment map, not as a final model description. The router, block size, pooling rule, fallback path, index dimension, GQA grouping, and reuse policy are all candidates for ablation. If the paper starts by presenting one fixed model, it hides the real scientific question. The correct framing is that the architecture itself is under test.

The current study keeps the dataset fixed to the real `speedrun_40M` subset and uses the same token budget for every run so that differences are easier to attribute to attention design rather than data volume.

4 Experimental Design

We organize the work along three axes.

Scale. Three model sizes: the current 5M-class model, a roughly 1.5× larger model, and a roughly 2.25× larger model. This checks whether the sparse mechanism survives capacity changes instead of only working at one size.

Context. Training uses a 256-token sequence length, while evaluation is repeated at 256, 512, and 1024 tokens. This checks whether the routing pattern changes when the prefix gets longer, which is the main reason to study sparse attention at all.

Architecture. The architecture is treated as the object of study, not as a fixed assumption. We compare dense attention with sparse variants that differ in block size, top-k, pooling, router projection sharing, local fallback, and reuse.

The smoke runs in this note use AdamW with learning rate 3×10^{-4}, weight decay 0.1, gradient clipping at 1.0, batch size 4, and an 80k-token train budget per run. Evaluation is repeated with four validation batches per context length. Those settings are deliberately small so contributors can reproduce and modify runs quickly.

5 Ablation Families

The paper is built around a small set of ablations that answer different questions.

Routing strength. Vary top-k and block size. This tests whether the router should read a few large regions or more smaller ones.

Router scoring. Vary the pooling operator used by the index branch, for example max, mean, or log-sum-exp. This tests what summary statistic is best for deciding which block survives.

Router capacity. Vary index dimension. This tests whether the router is underpowered, overpowered, or just noisy.

Projection sharing. Compare shared versus separate router/content projections. This tests whether the router should be derived from the attention stream itself or from a separate learned path.

Fallback behavior. Add a local fallback window plus routed blocks. This tests whether sparse attention still needs a dense safety net near the current position.

Reuse. Test cross-layer index reuse. This tests whether routing decisions are stable enough to share across depth instead of recomputing everything each layer.

Task stress. Add long-context retrieval and needle-style probes. This tests whether a sparse router preserves the blocks that matter when the answer is far back in the context.

6 Open-Source Research Plan

The project should be contributor-friendly because each useful contribution can be scoped to one proof obligation. The best issues are not "improve sparse attention". They are concrete checks or ablations with visible artifacts: a failing/passing test, a plot, a JSON result file, or a paper table update.

The first contributor tasks should therefore focus on:

- verifying the implementation against the reference mechanism;
- adding dense-oracle diagnostics for selected-block recall;
- adding a long-context retrieval benchmark where routing quality matters;
- testing hybrid local-plus-routed attention;
- expanding the router ablation matrix with index dimension and pooling variants;
- connecting the paper explanation more directly to the reference image;
- improving reproducibility by making every run produce a manifest, command, seed, and plot.

This gives contributors a reason to stay: each task is small enough to finish, but each one changes the research conclusion. The repository should reward contributions that make claims more checkable, not just contributions that add code.

7 What Is Under Test

Only after the design axes are clear do we state the attention family itself.

The implementation is a compact PyTorch sparse-attention block inspired by the diagrammed MiniMax structure: an index branch scores KV blocks, applies causal pooling, and selects the top-k blocks; the sparse branch then attends only over the selected blocks. The code is intentionally unfused so the paper can focus on architecture and experimental behavior before any kernel work.

7.1 Components

The attention family has four separable pieces.

Base transformer shape. The decoder scaffold: embedding width, number of layers, head count, KV head count, and feedforward width. This is the capacity axis.

GQA backbone. Grouped-query attention reduces KV cost while preserving multi-head query structure. This is the backbone the sparse mechanism sits on.

Index branch. The lightweight router scores blocks and selects the top-k candidates for each query group. This is the part that decides which memory survives.

Sparse branch. The content branch attends only over the selected blocks. Different pooling rules, local fallback behavior, and reuse strategies can all be plugged into this branch family.

8 Related Work

Sparse attention has several nearby ancestors. Sparse Transformer introduced hand-designed sparse factorizations and showed that structured sparsity can extend autoregressive modeling to much longer contexts than dense attention [2]. Longformer and BigBird combined local windows with global or random links for long-document modeling, and BigBird added theoretical coverage guarantees [3, 4]. Reformer and Routing Transformer took a more content-based route, using locality-sensitive hashing or online clustering to group similar tokens together [5, 6].

Grouped-query attention is orthogonal but important here: it reduces key/value cost without making attention sparse, which is why our sparse block sits on top of a GQA backbone rather than replacing it [7]. Recent production-scale work moves closer to our setup. MoBA routes queries to a few KV blocks and reports that block size and router design matter; NSA combines coarse token compression with finer token selection; and IndexCache shows that sparse index decisions can sometimes be reused across layers with little quality loss [8, 9, 10].

9 Results

Table 1 summarizes the two key runs so far.

Table 1: Real-data smoke results.

Model	Tokens	Seconds	Val loss	Val ppl	Tok/s
500k dense	501,760	88.49	7.4787	1769.88	5,650
500k sparse	501,760	69.11	7.4860	1782.89	7,234
1M dense	1,001,472	92.00	6.8665	959.54	10,886
1M sparse	1,001,472	127.24	6.8866	979.05	7,871

Figure 2 shows the 500k-token comparison and Figure 3 shows the longer 1M-token run with tagged panels.

10 Additional Sweep Results

We then ran the study sweep across three model scales and three evaluation context lengths. The main takeaways are simple: larger models help more than router tweaks, and the 1024-token evaluation does not collapse relative to 256 in this small-budget regime.

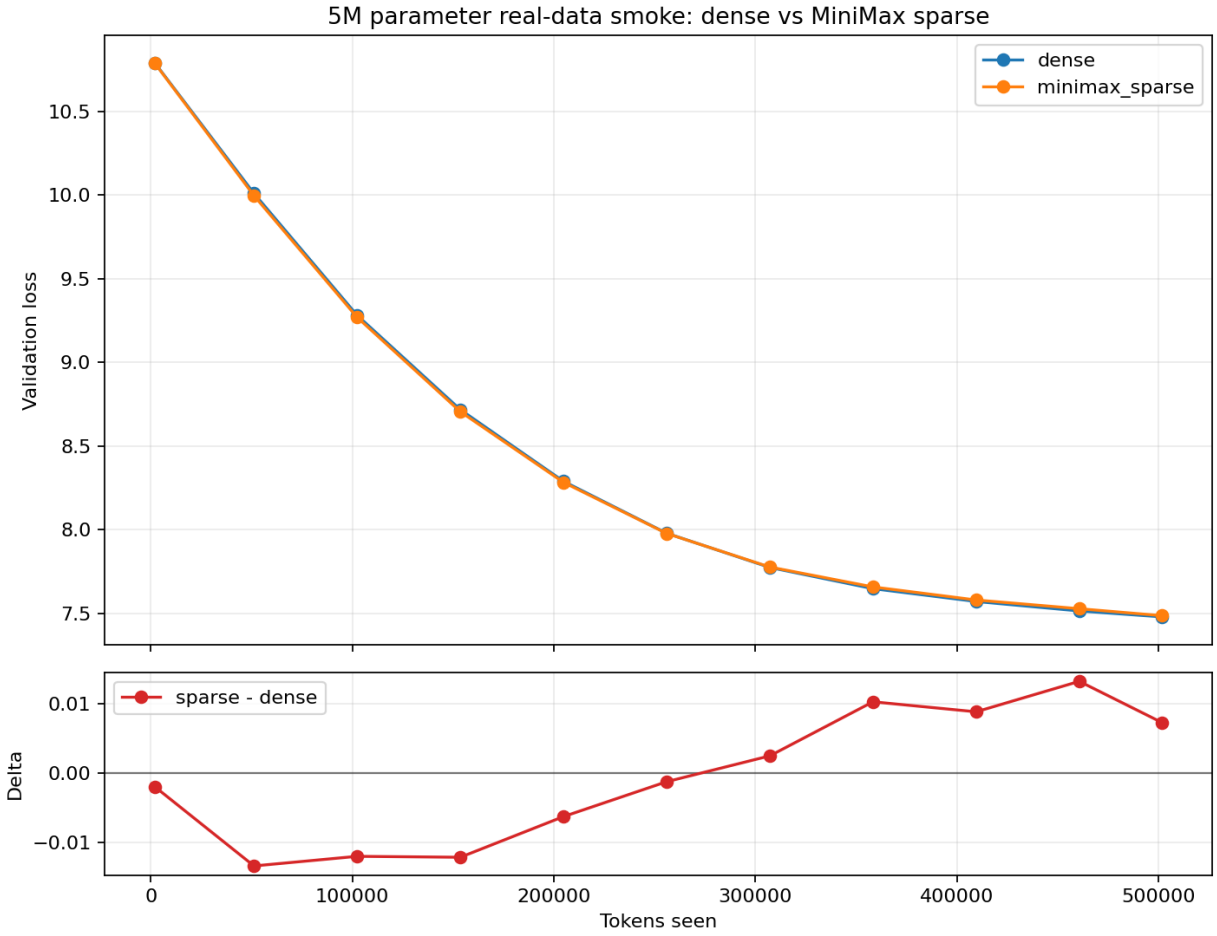

Figure 2: 500k-token dense vs sparse smoke run. The curves look close because the loss gap is small relative to the full loss range.

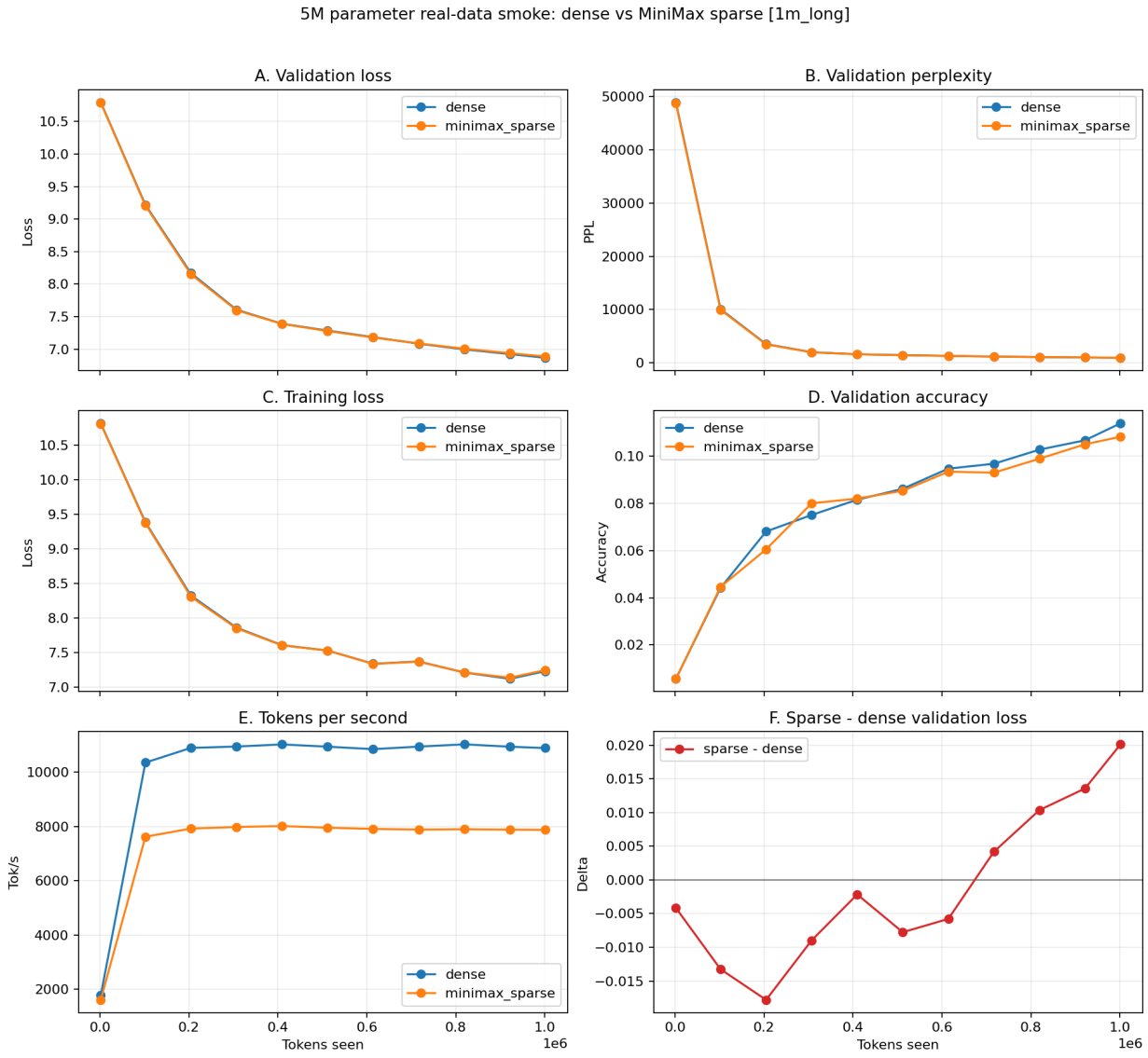

Figure 3: 1M-token dense vs sparse run. The delta panel shows the sparse model stays close but does not beat dense on validation loss in this run.

Figure 4 summarizes the scale/context sweep. Across the current, mid, and large settings, dense remains slightly ahead of sparse on the same token budget, but the gap stays small and the longer-context evaluation remains stable.

The more interesting architectural signal came from the mid-scale router sweep. Pooling style and shared-router projection barely changed the selected blocks on a held-out real batch, so the more diagnostic knobs were block size and top-k.

Figure 5 shows the matching architecture-ablation curves.

11 Analysis

The main result is not that sparse attention wins. It does not, at least not yet. Instead, the current evidence says:

- The sparse router is real and changes the computation path.
- On ordinary next-token loss, dense and sparse remain very close at this scale.
- The sparse model is not obviously better on quality, and throughput depends on the backend.
- The delta is small enough that plotting only a single loss curve is misleading.
- Pooling style and shared-router projection were too weak to separate cleanly at this budget; top-k and block size were the better architectural knobs.

The attention diagnostics also show the sparse path differs from dense in logits and routing density, so the near-overlap in training curves is not an implementation collapse. That makes this a good setup for ablations, not a dead end.

12 Open Questions and Ablations

The current runs say dense and sparse are close on short next-token loss, so the main question is no longer whether the router exists. The question is where the router helps, where it misses, and how much of the gain is routing quality versus backend efficiency.

The most informative follow-up experiments on this MacBook are:

- vary top-k at fixed block size
- vary block size at fixed top-k
- vary index dimension
- vary the pooling operator used by the index branch, for example max, mean, or log-sum-exp
- vary the number of KV heads and the query-to-KV grouping ratio
- compare shared versus separate projections for the router and the content branch
- add a local fallback window plus routed blocks to test whether near-context continuity matters
- test cross-layer index reuse, inspired by IndexCache, to see whether router outputs are stable enough to cache
- compare hard top-k selection with a softer or temperature-controlled router

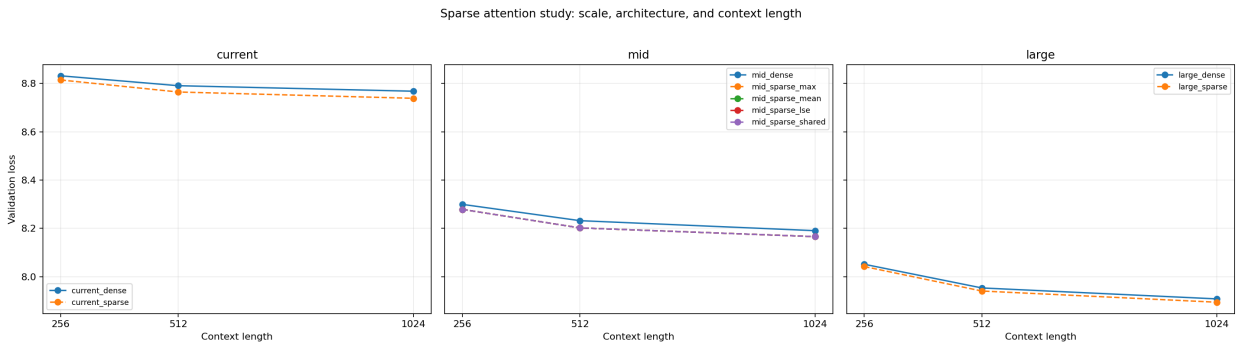

Figure 4: Scale and context sweep. Each panel shows validation loss versus context length for one model scale.

Table 2: Mid-scale sparse-attention architecture ablation on the real 40M slice. Density and selected-block counts are probe measurements on a held-out real batch.

Variant	Val loss	Tok/s	Probe density	Mean selected blocks
top-$k = 1$	8.2111	4896	0.2288	1.00
top-$k = 2$	8.2051	4764	0.3742	1.94
top-$k = 4$	8.2023	4202	0.5860	3.62
block size 8	8.2052	5226	0.2290	1.97
block size 32	8.2076	4440	0.5771	1.88

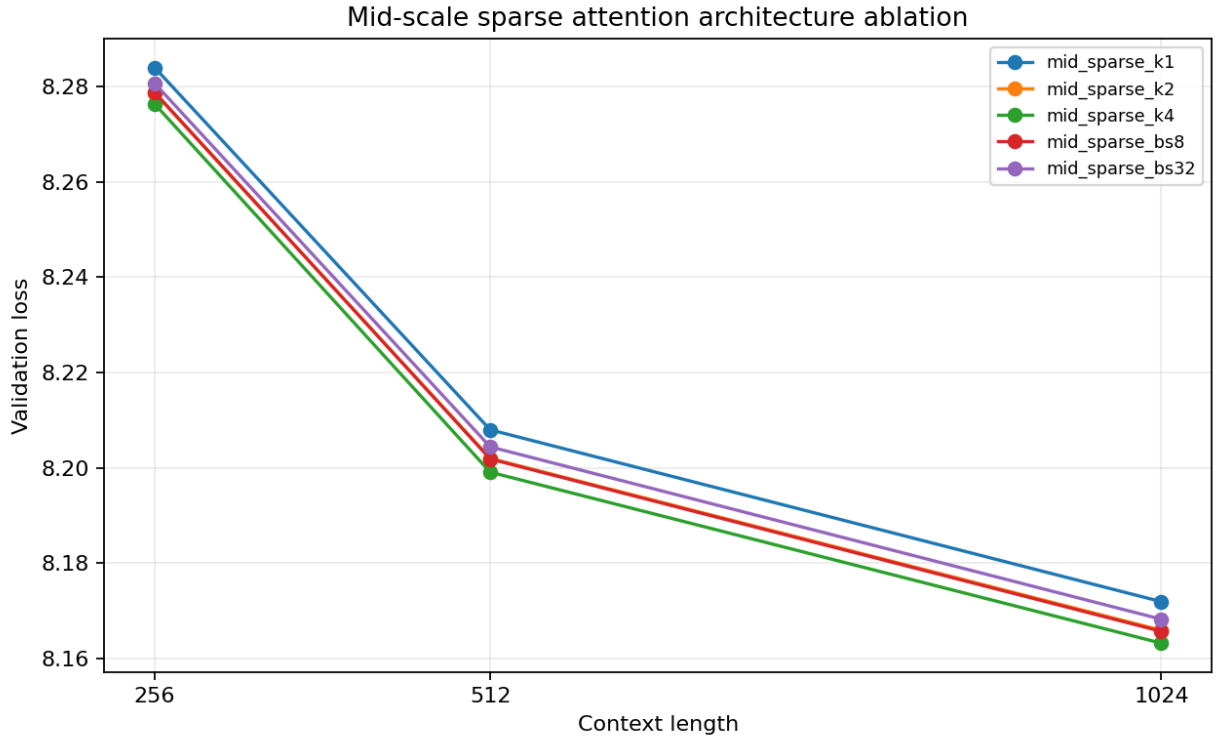

Figure 5: Mid-scale architecture ablation. Top-k and block size change the routing density and the final loss more than pooling or shared-router projection did in this run.

- compare dense vs sparse across three seeds
- add a long-context retrieval task where the queried answer sits many blocks earlier

For each ablation, the useful measurements are validation loss, perplexity, throughput, selected-block density, router entropy, selected-block recall against a dense oracle probe, and long-context retrieval accuracy.

13 Limitations

This is still a small-scale, unfused PyTorch study. It does not yet test the very long context regime from the MiniMax diagram, and it does not isolate whether any gain comes from routing quality or from backend-specific effects. That is fine for a first pass, but it means we should treat the current result as a research scaffold rather than a conclusion.

14 Conclusion

The project is in the useful middle state: the baseline is real, the sparse architecture is real, the plots are real, and the differences are measurable but modest. The next paper update should be about ablation curves, routing diagnostics, and a retrieval benchmark, not another generic next-token smoke run.

References

[1] Ashish Vaswani et al. Attention Is All You Need. https://arxiv.org/abs/1706.03762

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. https://arxiv.org/abs/1904.10509

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. https://arxiv.org/abs/2004.05150

[4] Manzil Zaheer et al. Big Bird: Transformers for Longer Sequences. https://arxiv.org/abs/2007.14062

[5] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. https://arxiv.org/abs/2001.04451

[6] Aurko Roy et al. Efficient Content-Based Sparse Attention with Routing Transformers. https://arxiv.org/abs/2003.05997

[7] Joshua Ainslie et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. https://arxiv.org/abs/2305.13245

[8] Enzhe Lu et al. MoBA: Mixture of Block Attention for Long-Context LLMs. https://arxiv.org/abs/2502.13189

[9] Jingyang Yuan et al. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. https://arxiv.org/abs/2502.11089

[10] Yushi Bai et al. IndexCache: Accelerating Sparse Attention via Cross-Layer Index Reuse. https://arxiv.org/abs/2603.12201