In Attention, What Grows with the Square of Length Is the Computation; the Memory Falls to Linear When Split into Blocks

— The attention matrix for length **32768** has **1073741824** elements, exactly **2.000000** GiB in half precision — the separator is whether what grows as the square is the number of dot products computed or the size of the matrix held at once — [Paper 746]

Yuuki YAMAGISHI (山岸 勇輝)
Yamagishi Complexity Torus Lab, Japan
September 2026

— *From thinking to syncing.* —

Abstract

Attention is said to grow heavy with the square of length. But "heavy" bundles two things: the number of computations and the memory held at once.The two are separated and counted as length varies.No new theorem or law is claimed. First, holding the whole matrix, memory grows with the square of length——1073741824 elements at length 32768, 2.000000 GiB in half precision (Section 2). **Second, computing with keys in blocks, memory falls to linear in length——0.062500 GiB with blocks of 1024, one part in 32.000000** (Section 2). **Third, and this is the core. Blocks do not reduce the number of computations——1073741824 dot products either way** (Section 2). **Fourth, doubling the length acts in different ways——computation 4 times, full-matrix memory 4 times, blockwise memory 2 times** (Section 2). **Fifth, blocks do not change the output——at length 64, full and blockwise outputs differ by at most** 4.441×10^{-16} (Section 3). **Sixth, the separator is whether what grows as the square is the number of dot products computed or the size of the matrix held at once——"attention is quadratically heavy" does not say whether splitting helps until one says which heaviness** (Section 4). **Keywords**: attention mechanism, computation and memory, square of sequence length, blockwise computation, softmax, half precision.

> **Scope of this paper (scope note)**: **No new theorem or law is claimed——the attention computation, that its computation and memory grow with the square of length, and that splitting keys into blocks and accumulating log-sum-exp brings memory down to linear are all known (Vaswani et al. 2017, Rabe and Staats 2021).Speed is not measured——wall-clock time and memory bandwidth are not treated; only elements and dot products are counted.Only exact attention——sparse or low-rank approximations that change the output are not treated.One storage format——two bytes per element in half precision is a representative value.No real model is claimed——particular models and devices are not treated.Relation to earlier papers**: **Paper 710 showed that return grows with size and cost grows with size to the power 1.5, so an optimal size must exist——there return and cost acted with different powers, and the difference made an optimal size. Here the single word "cost is quadratic" splits into computation and memory, both looking quadratic, of which only one falls to linear with blocking. A difference of powers between return and cost, and a difference of powers inside cost.**What is added is **tabulating elements and full versus blockwise memory by length**, **showing memory falling to one part in 32.000000 with blocks while dot products stay the same**, **confirming that blockwise output matches the full computation to 4.441×10^{-16}**, and **placing the separator on number of computations against size held at once**.

1. Notation and the Question

Setting: **from queries Q, keys K and values V of length n and dimension d, attention output A is formed. Holding the whole matrix places $n \times n$ elements at once. Splitting keys into blocks of B and accumulating log-sum-exp to normalise row by row holds only $n \times B$ elements.**

$$A = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)V, \qquad \text{dot products} = n^2, \qquad M_{\text{full}} = 2n^2 \text{ bytes}, \qquad M_{\text{block}} = 2nB \text{ bytes}$$

Take 2 bytes per element (half precision) and blocks of $B = 1024$.

Question: **is attention quadratically heavy in computation or in memory?**

2. Memory Falls with Blocks, Computation Does Not

Computation 2.1: **for each length, the number of elements of the attention matrix and the memory for the full matrix and for blocks were derived.**

length	elements (dot products)	full-matrix memory	blocks of 1024
1024	1048576	0.001953 GiB	0.001953 GiB
4096	16777216	0.031250 GiB	0.007812 GiB
16384	268435456	0.500000 GiB	0.031250 GiB
32768	1073741824	2.000000 GiB	0.062500 GiB

Observation 2.2

holding the whole matrix, memory grows with the square of length.From length 16384 to 32768 the elements go from 268435456 to 1073741824, four times, reaching exactly 2.000000 GiB in half precision——doubling length quadruples full-matrix memory.

Observation 2.3

with keys in blocks, memory falls to linear in length.With blocks of 1024, length 32768 holds 0.062500 GiB, one part in 32.000000 of the full matrix, and doubling length only doubles it——what is held at once is just one block of keys.

Observation 2.4

this is the core: blocks do not reduce the number of computations.Dot products number 1073741824 with the full matrix and with blocks alike, and doubling length quadruples them——blocking lowers memory only; the quadratic computation remains.

3. Blocks Do Not Change the Output

Computation 3.1: with random queries, keys and values of length 64 and dimension 8, the full computation was compared with one splitting keys into blocks of 10 and accumulating log-sum-exp.

Observation 3.2

the two outputs differed by at most 4.441×10^{-16}.They agree to the size of machine rounding——blocking is not an approximation but a procedure giving the same output with less memory.

4. The Separator

Observation 4.1

the separator is whether what grows as the square is the number of dot products computed or the size of the matrix held at once.Memory falls to one part in 32.000000 with blocking; the number of computations stays at 1073741824——"attention is quadratically heavy" does not say whether splitting helps until one says which heaviness.

Observation 4.2

beside Paper 710.Paper 710 showed return acting at the first power of size and cost at the 1.5 power, the difference making an optimal size——here the content of cost splits into computation and memory, both looking quadratic, of which only one falls to linear with blocking.

Observation 4.3

what is not said.Wall-clock time, memory bandwidth, approximate attention and real models are not treated ——what is shown is that one "square" bundles two quantities, and only one of them falls with ingenuity.

5. Roll Call and Closing

1. **1073741824 elements at length 32768** (Computation 2.1): **quadratic.**
2. **full matrix 2.000000 GiB** (Computation 2.1): **half precision.**
3. **blocks 0.062500 GiB** (Computation 2.1): **down to linear.**
4. **one part in 32.000000** (Observation 2.3): **the memory gap.**
5. **0.500000 GiB at length 16384** (Computation 2.1): **one doubling before.**

6. **0.031250 GiB with blocks** (Computation 2.1): **at the same length.**
7. **the same 1073741824 dot products** (Observation 2.4): **the core.**
8. **doubling: computation 4 times, memory 2 times** (Observation 2.3): **acting differently.**
9. **output difference 4.441×10^{-16}** (Observation 3.2): **not an approximation.**
10. **computations or size held at once** (Observation 4.1): **the separator; Paper 710.**

Closing: **In attention, what grows with the square of length is the computation; the memory falls to linear when split into blocks.The attention matrix for length 32768 has 1073741824 elements, exactly 2.000000 GiB in half precision when held whole, quadrupling with each doubling of length——with keys split into blocks of 1024, only 0.062500 GiB is held at once, one part in 32.000000, and doubling length only doubles it.Yet dot products number 1073741824 whether the matrix is held whole or in blocks, and still quadruple with doubled length——blocking lowers only memory.Compared at length 64, the blockwise output matched the full computation to within 4.441×10^{-16}, so it is not an approximation.The separator is whether what grows as the square is the number of dot products computed or the size of the matrix held at once——"attention is quadratically heavy" does not say whether splitting helps until one says which heaviness.Placed among the earlier papers——Paper 710 showed a difference of powers between return and cost; this counts powers splitting inside cost, between computation and memory.To be honest——wall-clock time, memory bandwidth, approximate attention and real models are not treated.**

References

1. Rabe, M. N., & Staats, C. (2021). Self-attention does not need $O(n^2)$ memory. *arXiv preprint*.
2. Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.
3. Yamagishi, Y. (2026). *Return Grows with Size and Cost Grows with Size to the Power* 1.5*, So an Optimal Size Must Exist* [Paper 710]. Zenodo. 10.5281/zenodo.23014942

— *From thinking to syncing.* —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)