A Note on 2-simplical attention versus dot product attention

by Michael Malak, 7/4/2025

What if every pairwise signal inside a Transformer suddenly gained a third witness -- and your GPU(s) barely flinched?

Researchers from Meta, in "Fast and Simplex: 2-Simplicial Attention in Triton" ([2]), swap the dot-product edge for a triangle, letting each head weigh ordered triplets instead of pairs. A custom kernel written in Triton (OpenAI's Python framework that simplifies CUDA) keeps throughput within a hair of Flash-Attention, yet on a 3.5 B-parameter Mixture-of-Experts the switch trims GSM8k negative-log-likelihood from 0.2781 to 0.2718 (–2.3%) and steepens the scaling-law exponent α from 0.142 to 0.168. Those decimals look tame until you recall that perplexity is exponential; each percent often converts to several exact-match points after decoding. And because compute cost barely budges, the triangle’s gain is practically free.

A few days ago, I posted about Joshi's paper showing that the attention matrix can be viewed as a complete graph. To picture 2-simplicial attention in that language, you simply upgrade the graph to a hypergraph, where every hyperedge links three nodes instead of two. Joshi used Graph Attention Networks (GAT) to interpret pairwise heads; moving to triplets naturally evokes Hypergraph Attention Networks (HyperGAT), coined by Bai et al. in 2019 ([6]). HyperGAT’s recipe is: gather the features of all nodes in a hyperedge, compute an attention weight for each, combine them, then send the result back to every member. 2-simplicial attention performs the same gather-and-scatter routine, but bakes it straight into the Transformer's kernel and, because it keeps the triangle's edges implicitly alive, still enjoys the algebraic conveniences (homology, spectral tricks) that pure hypergraphs can lose.

Think of the progression like this: GAT lets every pairwise edge decide how much weight its two tokens share; HyperGAT lets an entire multi-node group negotiate a joint message; 2-simplicial attention hard-wires the simplest non-trivial group -- a triangle -- into the kernel itself, so three tokens can talk together natively, adding virtually no extra latency.

But the real battle isn't speed versus quality; it's voracious models versus a shrinking buffet of clean text. Classic dot-product attention gorges on ever more tokens. 2-Simplicial attention hoards information, mining three-way correlations from the same morsels.

If triangles are only step one, are we heading for an arms race (tetrahedra, pentachora, etc.) over how many tokens attention can juggle at once?

A diagram of a diagram of a product attention and a diagram of a product attention

Description automatically generated

# References

[1] [original Linkedin post by M. Malak, 7/4/2025](https://www.linkedin.com/posts/michaelmalak_what-if-every-pairwise-signal-inside-a-transformer-activity-7346886605783113728-SXev?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAFZfUoBgPoGUucdnvtwuzPv79P8VHj6uvk)

[2] [Fast and Simplex: 2-Simplicial Attention in Triton, Aurko Roy et al, Meta, 2025](https://github.com/dimitarpg13/large_language_models/blob/main/articles/attention/Fast_and_Simplex-2-Simplicial_Attention_in_Triton_Roy_Meta_2025.pdf)

[3] [Introducing Triton, OpenAI, July 28, 2021](https://openai.com/index/triton/)

[4] <https://github.com/triton-lang/triton>

[5] <https://triton-lang.org/main/index.html>

[6] [Hypergraph Convolution and Hypergraph Attention, Song Bai et al, U. of Oxford, 2020](https://github.com/dimitarpg13/deep_learning_and_neural_networks/blob/main/literature/articles/graph_networks/Hypergraph_Convolution_and_Hypergraph_Attention_Bai_2020.pdf)