Fast and Simplex: 2-Simplicial Attention in Triton

Rethinking Transformer Attention with Trilinear Forms

Amit Kumar

AI/NLP Engineer at E42.ai

Table of contents

The Token Efficiency Challenge

- **Scaling Laws Problem:** Current scaling laws assume infinite high-quality data
  - Traditional: $L(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}$
  - Reality: High-quality tokens are becoming scarce
- **Compute vs Data Bound:** Models are shifting from compute-bound to data-bound
  - Internet-scale datasets have limitations
  - Need architectures that prioritize token efficiency
- **Architecture Innovation:** Most modifications only shift the constant, not the exponent
  - Goal: Change the scaling law exponent itself
  - Improve performance under token constraints

Standard Dot-Product Attention (1-Simplex)

Standard Transformer Attention:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{1}$$

$$A_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d}} \quad \text{(Bilinear form)} \tag{2}$$

$$S_{ij} = \frac{\exp(A_{ij})}{\sum_{j=1}^{n} \exp(A_{ij})} \quad \text{(Row-wise softmax)} \tag{3}$$

$$\tilde{v}_i = \sum_{j=1}^{n} S_{ij} v_j \quad \text{(Weighted sum)} \tag{4}$$

- **Complexity:** $O(n^2)$ for sequence length n
- **Geometric View:** Pairwise relationships (1-simplex)

Geometry of 2-Simplicial Attention

(a) 1-simplex between two nodes i, j (b) 2-simplex between three nodes i, j, k

Figure 1: Geometry of dot product attention and 2-simplical attention.

2-Simplicial Attention: From Bilinear to Trilinear

Key Innovation: Extend from dot-product (bilinear) to trilinear forms

Standard Attention (1-simplex):

- Two matrices: Q, K
- Pairwise interactions
- $A_{ij} = \langle q_i, k_j \rangle$

2-Simplicial Attention (2-simplex):

- Three matrices: Q, K, K'
- Triplet interactions
- $A_{ijk} = \langle q_i, k_j, k'_k \rangle$

Geometric Interpretation:

- 1-simplex: Line segment between two points
- 2-simplex: Triangle connecting three points
- Higher-order relationships capture complex dependencies

2-Simplicial Attention Mathematical Formulation

Extended Projections:

$$K' = XW_{K'}, \quad V' = XW_{V'} \quad \text{(Additional key-value pairs)} \tag{5}$$

Trilinear Attention Logits:

$$A_{ijk}^{(2s)} = \frac{\langle q_i, k_j, k'_k \rangle}{\sqrt{d}} = \frac{1}{\sqrt{d}} \sum_{l=1}^{d} Q_{il} K_{jl} K'_{kl} \tag{6}$$

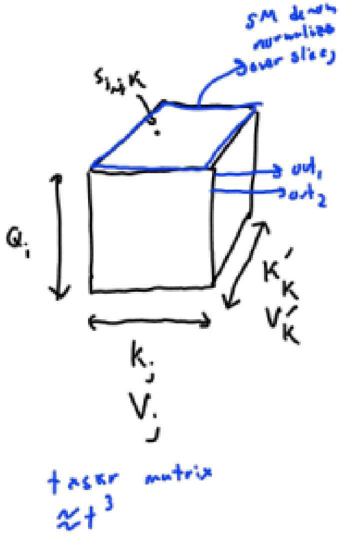
2D Softmax:

$$S_{ijk}^{(2s)} = \frac{\exp(A_{ijk}^{(2s)})}{\sum_{j,k} \exp(A_{ijk}^{(2s)})} \tag{7}$$

Output with Hadamard Product:

$$\tilde{v}_i^{(2s)} = \sum_{j,k=1}^{n} S_{ijk}^{(2s)} (v_j \circ v'_k) \tag{8}$$

Simplified diagram



The RoPE Compatibility Problem

Issue: Trilinear forms are NOT rotation invariant

- **Standard Attention:** $\langle Rq_i, Rk_j \rangle = \langle q_i, k_j \rangle$
- **Trilinear Form:** $\langle Rq_i, Rk_j, Rk'_k \rangle \neq \langle q_i, k_j, k'_k \rangle$

Why This Matters:

- RoPE (Rotary Position Embedding) requires rotational invariance
- Position information would be corrupted
- Cannot directly apply existing positional encodings

Solution: Use determinant-based trilinear forms instead!

Determinant-Based Trilinear Forms

Key Insight: Determinants ARE rotation invariant

$$f_3(a,b,c) = \det \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} \tag{9}$$

Sarrus Rule Expansion:

$$\begin{aligned} f_3(a,b,c) &= a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_1b_3c_2 - a_2b_1c_3 - a_3b_2c_1 && (10) \\ &= \langle (a_1,a_2,a_3),(b_2,b_3,b_1),(c_3,c_1,c_2) \rangle && (11) \\ &\quad - \langle (a_1,a_2,a_3),(b_3,b_1,b_2),(c_2,c_3,c_1) \rangle && (12) \end{aligned}$$

Computational Cost: 2 dot products instead of 1
Geometric Meaning: Volume of parallelepiped (rotation invariant!)

Computational Complexity and Windowing

Complexity Challenge:

- Naive 2-simplicial attention: $O(n^3)$ - impractical!
- Standard attention: $O(n^2)$

Sliding Window Solution:

$$O(\text{2-simplicial}) = 6nw_1w_2 \quad (\text{instead of } 6n^3) \tag{13}$$

- Each query q_i attends to:
  - w_1 keys from K (window size 1)
  - w_2 keys from K' (window size 2)
- Chosen configuration: $(w_1, w_2) = (512, 32)$
- Computational cost comparable to 48k context standard attention

Triton Kernel Implementation

Flash Attention Extension:

- Built on Flash Attention principles
- Online softmax for memory efficiency
- 2D tiling for trilinear operations

Key Optimizations:

1. **Elementwise Multiplication:** Merge QK via element-wise multiply
2. **Tensor Core Utilization:** $(QK) \otimes K'$ and $P \otimes (VV')$
3. **Overlap Computation:** CUDA Core + Tensor Core parallel execution
4. **Two-Stage Backward:** Separate kernels for dK, dV and dK', dV', dQ

Model Architecture and Training

Experimental Setup:

- **Model Type:** Mixture of Experts (MoE)
- **Size Range:** 1B to 3.5B active parameters
- **Total Parameters:** 57B to 176B total parameters
- **Architecture:** Every 4th layer uses 2-simplicial attention

Training Configuration:

- **Optimizer:** AdamW with peak LR 4×10^{3}
- **Weight Decay:** 0.0125
- **Schedule:** 4000 step warmup + cosine decay
- **Evaluation:** GSM8k, MMLU, MMLU-pro, MBPP

Focus Benchmarks: Mathematics, reasoning, and coding tasks where 2-simplicial attention shows strongest benefits

Performance Results: Negative Log-Likelihood

Model	Active Params	GSM8k	MMLU	MMLU-pro	MBPP
Transformer	1B	0.3277	0.6411	0.8718	0.2690
2-simplicial	1B	0.3302	0.6423	0.8718	0.2714
$\Delta(\%)$		+0.79%	+0.19%	-0.01%	+0.88%
Transformer	2B	0.2987	0.5932	0.8193	0.2435
2-simplicial	2B	0.2942	0.5862	0.8135	0.2411
$\Delta(\%)$		-1.51%	-1.19%	-0.71%	-1%
Transformer	3.5B	0.2781	0.5543	0.7858	0.2203
2-simplicial	3.5B	0.2718	0.5484	0.7689	0.2193
$\Delta(\%)$		**-2.27%**	**-1.06%**	**-2.15%**	**-0.45%**

Key Observations:

- Gains increase with model size (scaling benefits)
- Strongest improvements on reasoning tasks (GSM8k, MMLU-pro)
- Minimal gains for models $<$ 2B parameters

Scaling Law Analysis

Loss Function: $L(N) = E' + \frac{A}{N^\alpha}$

Model	**GSM8k** α	**MMLU** α	**MMLU-pro** α	**MBPP** α
Transformer	0.1420	0.1256	0.0901	0.1720
2-simplicial	0.1683	0.1364	0.1083	0.1837
$\Delta(\%)$	**+18.5%**	**+8.5%**	**+20.2%**	**+6.8%**

Critical Finding: 2-simplicial attention achieves **steeper scaling slopes**

- Higher α means better parameter efficiency
- Largest improvements on challenging benchmarks (MMLU-pro: +20.2%)
- Enables more favorable scaling under token constraints
- All fits show excellent $R^2 > 0.99$

Main Contributions and Impact

1. **Architectural Innovation:**
   - First practical implementation of 2-simplicial attention
   - Generalizes from bilinear to trilinear attention forms
   - Maintains computational tractability through windowing
2. **Theoretical Advances:**
   - Solved RoPE compatibility with determinant-based forms
   - Proved expressivity advantages for complex reasoning tasks
   - Demonstrated scaling law exponent improvements
3. **Efficient Implementation:**
   - High-performance Triton kernel (520 TFLOPS)
   - Memory-efficient sliding window attention
   - Competitive with Flash Attention v3
4. **Empirical Validation:**
   - Better token efficiency on reasoning tasks
   - Improved scaling exponents (up to +20.2%)
   - Validated on models up to 176B total parameters

FAv3 vs 2-simplicial attention

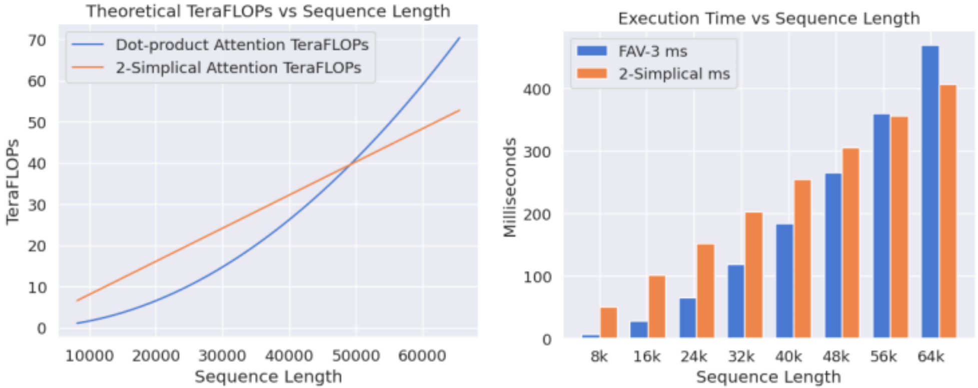


Figure 3: FLOPs and Latencies of FAv3 vs 2-simplical attention

Conclusion

- **Token Efficiency:** 2-simplicial attention provides a path toward better scaling under data constraints
- **Scaling Law Innovation:** First architecture to meaningfully change scaling exponents, not just constants
- **Reasoning Benefits:** Particularly strong on mathematics, coding, and logical reasoning tasks
- **Implementation Ready:** Efficient Triton kernels make this practical for real deployments

Future Directions

Future Work:

- Optimize kernel implementation further
- Explore higher-order simplicial attention (3-simplex, 4-simplex)
- Investigate optimal window sizes for different modalities
- Scale to larger models and more diverse tasks

Key Takeaway: In a data-constrained world, architectural innovations like 2-simplicial attention offer promising paths to better token efficiency and improved scaling laws.

Thank You