SCALING LAWS FOR HIGHER-ORDER ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model scaling relies on ever more training data, and treats per-token computation as a cost to minimize. As high-quality data runs out, we ask whether *increasing* per-token computation can improve performance instead. We compare scaling laws of standard and higher-order attention, where each query attends jointly over pairs of keys, making attention cubic rather than quadratic in sequence length. To enable this, we release dense and dilated higher-order-attention Triton kernels with ALiBi3D positional encoding, supporting long-context training and length extrapolation. At matched parameters and data, higher-order attention attains lower loss; standard attention needs 15–30% more tokens to match it. The gain stems from a lower irreducible loss ($\Delta E \approx 0.134$ nats), not a faster data exponent, and carries over to downstream tasks. We then ask whether higher-order attention is *more efficient*. We find that it depends on the *resource*: in our regime, higher-order attention wins per parameter and per token, while standard attention wins per FLOP and per byte of KV cache. Because higher-order attention has a lower loss floor, our fitted laws do predict it overtakes standard attention per FLOP and per byte of KV cache beyond our training scale, signaling the promise of higher-order attention at large compute budgets.

1 INTRODUCTION

Scaling language models has relied on larger models and more training data (Kaplan et al., 2020; Hoffmann et al., 2022). Yet public, high-quality data is finite and nearing exhaustion (Villalobos et al., 2024), shifting the question from how much data to use toward how much computation to spend extracting value from data. The field treats per-token training compute as a cost to minimize: sparse Mixture-of-Experts models separate parameter count from per-token computation (Dai et al., 2024; Muennighoff et al., 2025); linear-attention and recurrent architectures reduce per-token computation (Katharopoulos et al., 2020; Gu & Dao, 2023; Team et al., 2025; Danieli et al., 2025); and chain-of-thought methods add computation at inference, not during training (Wei et al., 2022; Snell et al., 2025). We study the axis left open: spending more compute on every token *during training* while keeping parameter counts practical. This treats per-token compute as a third scaling axis alongside model capacity and training data.

Higher-order attention offers one way to scale per-token computation without proportionally increasing parameters. Where standard self-attention scores token pairs, higher-order attention (also called 2-simplicial, trilinear, or cubic attention) has each query attend jointly over pairs of keys, scoring token triples (Clift et al., 2019; Roy et al., 2025). These three-way scores can represent functions that standard attention cannot (Sanford et al., 2023; Kozachinskiy et al., 2025), and Clift et al. (2019); Roy et al. (2025) report gains in reasoning and language-model scaling. Yet those gains were measured either at small scale or with higher-order attention confined to windowed or interleaved layers. Applied densely at every layer, it costs $O(n_{ctx}^2)$ per token, cubic over the sequence, and caches twice the Key-Value (KV) state per token. It remains open whether going beyond pairwise interactions is worth this added compute and memory at scale. This motivates our central question:

When is higher-order attention worth its cost?

To answer it, we make dense higher-order attention practical to train, then fit scaling laws that compare it against standard attention separately under each resource that governs cost, as Dehghani et al. (2021) recommend. Our findings are as follows.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1. **Dense higher-order attention can be trained stably at scale.** Stabilized optimization, ALiBi3D positional encodings, and our released Triton kernels let models with higher-order attention in every layer train past one billion parameters, with loss holding at up to four times the training context length (Sections 3, 3.2 and 4 and Figure 3.1c).
2. **Higher-order attention reaches a given loss on fewer tokens.** At matched parameters and tokens, it attains 0.03–0.07 nats lower cross-entropy, so at fixed model size standard attention needs 15–30% more tokens to match it. The gain comes from a lower irreducible loss ($\Delta E \approx 0.134$ nats) rather than a faster data exponent (Section 5).
3. **Whether higher-order attention is *more efficient* depends on the resource counted.** It wins per parameter and per token at every scale we train, while standard attention wins per training FLOP and per byte of KV cache, since higher-order attention's KV cache is twice as large (Section 5.2). Because of its lower loss floor, our fitted laws predict higher-order attention *does overtake* standard attention per FLOP beyond our training scale ($C \approx 10^{23}$ at $n_{ctx} = 512$), with the break-even budget growing at longer contexts.
4. **Sparsity preserves the advantage at long context.** The dilated variant matches dense higher-order attention's loss with per-token cost linear rather than quadratic in n_{ctx}: at $n_{ctx} = 2048$, it costs 19× less than dense and 8× more than standard attention (Section 6 and Figure K.1).
5. **The loss advantage transfers to downstream tasks.** Across a suite of downstream benchmarks, both architectures fall on a single curve of accuracy against pretraining loss, so higher-order attention's lower loss translates quantitatively into higher accuracy (Section 7).

Together, these findings position per-token computation as a *third scaling axis*: worth its cost per parameter and per token at the scales we train, and expected to pay off per FLOP and per byte of KV cache beyond the crossover budget (Section 5.2).

2 Preliminaries

Throughout the paper, n_{ctx} is the sequence length, n_{vocab} the vocabulary size, d_{model} the residual-stream width and d_{head} the per-head width, with n_{head} heads so that $d_{model} = n_{head} d_{head}$. Positions are indexed by i (queries) and j_1, j_2 (keys), and channels within a head by $a \in \{1, \dots, d_{head}\}$. All head widths equal d_{head}; only query and key widths must match, as the score contracts them over a. Refer to Section A for detailed notation.

2.1 Language Modeling

We consider autoregressive next-token prediction over a vocabulary $\mathcal{V} = \{1, \dots, n_{vocab}\}$. Given a context $\boldsymbol{x}^{(<i)} = (x^{(1)}, \dots, x^{(i-1)})$, a network f with parameters $\boldsymbol{\theta}$ produces logits

$$\boldsymbol{z}^{(i)} = f(\boldsymbol{x}^{(<i)}; \boldsymbol{\theta}) \in \mathbb{R}^{n_{vocab}}, \tag{2.1}$$

whose softmax $\sigma_u(\boldsymbol{z}) = \exp(z_u)/Z(\boldsymbol{z})$, with partition function $Z(\boldsymbol{z}) = \sum_{w=1}^{n_{vocab}} \exp(z_w)$, defines the next-token distribution. We add a *z-loss* term for stability (Chowdhery et al., 2022),

$$\mathcal{L}\big(x^{(i)}, \boldsymbol{z}^{(i)}\big) = -\log \sigma_{x^{(i)}}(\boldsymbol{z}^{(i)}) + \lambda_Z \big(\log Z(\boldsymbol{z}^{(i)})\big)^2, \tag{2.2}$$

with $\lambda_Z = 10^{-4}$ throughout. The training objective sums Equation (2.2) over all tokens.

2.2 Standard (Quadratic) Attention

We restate the computation of a *single attention head* to fix notation for the higher-order case; the remainder of the attention block, the residual connection, and the MLP are standard (Vaswani et al., 2023). From an input sequence $\boldsymbol{X} \in \mathbb{R}^{n_{ctx} \times d_{model}}$, a head forms query, key, and value projections

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}^{Z} \quad \text{for } Z \in \{Q, K, V\}, \qquad \boldsymbol{W}^{Z} \in \mathbb{R}^{d_{model} \times d_{head}}. \tag{2.3}$$

The attention score *matrix* $\boldsymbol{S} \in \mathbb{R}^{n_{ctx} \times n_{ctx}}$ has entries $S_{i,j}$, each a scaled dot product of query i and key j; a softmax over j and a weighted sum of the values give the single head's output $O_{i,a}$,

$$S_{i,j} = \frac{1}{\sqrt{d_{head}}} \sum_{a=1}^{d_{head}} Q_{i,a} K_{j,a}, \qquad A_{i,j} = \frac{\exp(S_{i,j})}{\sum_{j'=1}^{n_{ctx}} \exp(S_{i,j'})}, \qquad O_{i,a} = \sum_{j=1}^{n_{ctx}} A_{i,j} V_{j,a}. \tag{2.4}$$

This is a *quadratic interaction* between two representations, costing $O(n_{ctx}^2)$ compute.

2.3 Higher-Order (Cubic) Attention

Higher-order attention (Clift et al., 2019; Roy et al., 2025) extends this quadratic interaction to three input representations: a query and two keys. A query attends to ordered pairs of positions rather than to individual positions. From the same input $\boldsymbol{X} \in \mathbb{R}^{n_{ctx} \times d_{model}}$, a single head forms

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}^{Z} \quad \text{for } Z \in \{Q, K^1, K^2, V^1, V^2\}, \qquad \boldsymbol{W}^{Z} \in \mathbb{R}^{d_{model} \times d_{head}}. \tag{2.5}$$

The score *tensor* $\mathbf{S} \in \mathbb{R}^{n_{ctx} \times n_{ctx} \times n_{ctx}}$ is a three-way contraction of a query with two keys,

$$S_{i,j_1,j_2} = \frac{1}{\sqrt{d_{head}}} \sum_{a=1}^{d_{head}} Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}. \tag{2.6}$$

As in Equation (2.4), the $\sqrt{d_{head}}$ normalizer holds the pre-softmax scores at unit variance.[1] A joint softmax over (j_1, j_2) and a weighted sum of the paired values give the output $O_{i,a}$,

$$A_{i,j_1,j_2} = \frac{\exp(S_{i,j_1,j_2})}{\sum_{j_1'=1}^{n_{ctx}} \sum_{j_2'=1}^{n_{ctx}} \exp(S_{i,j_1',j_2'})}, \qquad O_{i,a} = \sum_{j_1=1}^{n_{ctx}} \sum_{j_2=1}^{n_{ctx}} A_{i,j_1,j_2} V^1_{j_1,a} V^2_{j_2,a}. \tag{2.7}$$

The score and attention tensors are third-order, costing $O(n_{ctx}^3)$ compute.

2.4 Experimental Setting

All models are a pre-norm, decoder-only dense Transformer (Grattafiori et al., 2024) with the higher-order attention of Section 2.3. The Multi-Layer Perceptrons (MLPs) are Swish-Gated Linear Unit (SwiGLU) with a Feed-Forward Network (FFN) ratio of 2.75. Positions are encoded with ALiBi3D (Section 3.2). We optimize with Adam with decoupled weight decay (AdamW) and transfer the learning rate across model sizes with CompleteP (Dey et al., 2025; Mlodozeniec et al., 2025). We train on the high-quality split of Nemotron-CC (Su et al., 2025). We report cross-entropy on held-out data from the same split. Full hyperparameters are given in Section L; experiment-specific ones appear where introduced.

3 Dense Higher-Order Attention

Higher-order attention (Section 2.3) can replace the standard attention operator, but two parts of the usual transformer recipe do not carry over: (i) the product of two value streams couples their gradients and destabilizes training, which we address by normalizing V^1 and V^2 (Section 3.1); and (ii) rotary positional embeddings lose their relative-position structure in the three-way score, so we introduce ALiBi3D instead (Section 3.2). The score normalization, by contrast, carries over unchanged: we show that it matches standard attention's $\sqrt{d_{head}}$ at any interaction order (Proposition 2).

3.1 Normalizing Values Stabilizes Training

The value product $V^1_{j_1,\cdot} \odot V^2_{j_2,\cdot}$ in Equation (2.7) couples the gradients of V^1 and V^2. Standard attention has no analogue: its output is linear in V (Equation (2.4)), so the value gradient does not depend on V. In higher-order attention, differentiating Equation (2.7) through the value product gives:

$$\frac{\partial \mathcal{L}}{\partial V^1_{j_1,a}} = \sum_{i,\, j_2} A_{i,j_1,j_2} \frac{\partial \mathcal{L}}{\partial O_{i,a}} V^2_{j_2,a}, \tag{3.1}$$

and symmetrically for V^2. Each gradient grows with the other's magnitude, forming a positive feedback loop that drives the two norms to diverge. To break the loop, we apply a Root-Mean-Square Normalization (RMSNorm) to each of V^1, V^2, per token over d_{head}, with a learnable gain:

$$\widetilde{V}^c_{j,a} = \gamma_c \, V^c_{j,a} / \mathrm{RMS}(V^c_{j,\cdot}), \qquad \mathrm{RMS}(x) = \sqrt{\tfrac{1}{d} \textstyle\sum_{k=1}^{d} x_k^2}, \qquad c \in \{1, 2\}. \tag{3.2}$$

We use $\widetilde{V}^1, \widetilde{V}^2$ throughout. This gives $\mathbb{E}\|\widetilde{V}^1 \odot \widetilde{V}^2\|^2 = d_{head}\, \gamma_1^2 \gamma_2^2$, independent of the projection scale, so the feedback loop cannot form. Figure 3.1a compares the recipes: without normalization the projections diverge, with it training stays stable, and freezing one gain tightens the norms.

[1] We prove the same $\sqrt{d_{head}}$ holds at any order in Proposition 2.

3.2 ALiBi3D Extrapolates to Longer Contexts

Cubic attention is incompatible with RoPE. Rotary Position Embedding (RoPE) (Su et al., 2023) performs well in quadratic attention transformers because the rotated dot product depends only on relative position. It rotates the row $\boldsymbol{Z}_i \in \mathbb{R}^{d_{head}}$ at each position i by a position-dependent matrix $\boldsymbol{R}^{(i)}$, $\boldsymbol{Z}'_i = \boldsymbol{R}^{(i)}\boldsymbol{Z}_i$, for every query and key. In the quadratic case, the score $S_{i,j}$ of query i and key j depends only on the relative index $j - i$:

$$S_{i,j} \propto (\boldsymbol{Q}'_i)^\top \boldsymbol{K}'_j = \boldsymbol{Q}_i^\top \boldsymbol{R}^{(i)\top}\boldsymbol{R}^{(j)}\boldsymbol{K}_j = \boldsymbol{Q}_i^\top \boldsymbol{R}^{(j-i)}\boldsymbol{K}_j. \tag{3.3}$$

Cubic attention rotates the query $\boldsymbol{Q}_i$ and the two keys $\boldsymbol{K}^1_{j_1}, \boldsymbol{K}^2_{j_2}$ the same way, giving the score

$$S_{i,j_1,j_2} \propto \sum_{a=1}^{d_{head}} (\boldsymbol{R}^{(i)}\boldsymbol{Q}_i)_a (\boldsymbol{R}^{(j_1)}\boldsymbol{K}^1_{j_1})_a (\boldsymbol{R}^{(j_2)}\boldsymbol{K}^2_{j_2})_a. \tag{3.4}$$

The reduction in Equation (3.3) relied on the two rotations pairing as $\boldsymbol{R}^{(i)\top}\boldsymbol{R}^{(j)} = \boldsymbol{R}^{(j-i)}$. Here the three rotations meet at the contracted index a with no such pairing, so they do not combine into the relative rotations $\boldsymbol{R}^{(j_1-i)}, \boldsymbol{R}^{(j_2-i)}$. The score cannot be written as a function of the relative displacements $(j_1 - i, j_2 - i)$ alone; it retains a dependence on the absolute position i. The natural extension of RoPE loses the translation-equivariant relative-position structure that makes it effective. Roy et al. (2025) instead propose a determinant-based three-way score that is invariant to joint rotations of the three tokens. This invariance holds for the block-diagonal subgroup induced by their per-chunk construction, not for general orthogonal transformations (Section E.1).

ALiBi. Unlike RoPE, whose relative-position property depends on how the operator combines rotated vectors, Attention with Linear Biases (ALiBi) (Press et al., 2022) adds a bias to the logits after scoring, so translation invariance holds for any attention operator. ALiBi introduces positional information by adding a distance-dependent bias to the attention logits that is translation invariant and induces a monotone preference for nearby tokens. For query index i and key index j, with slope $m > 0$, these two properties read $b(i + t, j + t) = b(i, j)$ and $b(i, j) = -m|i - j|$.

ALiBi3D. For cubic attention we require the analogous property:

$$b(i + t, j_1 + t, j_2 + t) = b(i, j_1, j_2). \tag{3.5}$$

We introduce *ALiBi3D*, summing an ALiBi bias over each query-key pair:

$$b(i, j_1, j_2) = -m\left(|i - j_1|^\omega + |i - j_2|^\omega\right). \tag{3.6}$$

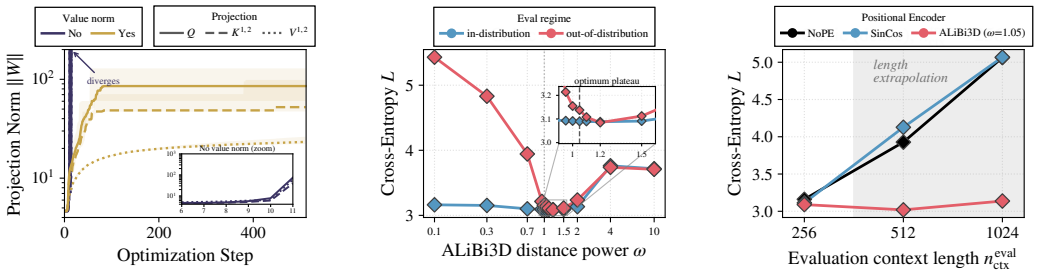
The exponent ω makes the distance penalty nonlinear. Each term depends only on a difference $i - j_k$ for $k \in \{1, 2\}$, so the bias satisfies Equation (3.5).

The linear distance penalty is degenerate. We sweep ω and evaluate both within and beyond the training context (Figure 3.1b, Section L). Within the training context, performance is largely insensitive to ω. Differences can be observed under length extrapolation, where values slightly above 1 perform best and the optimum lies below 1.5. In particular, the linear choice $\omega = 1$ is suboptimal for the reason discussed in Section E.2. We use $\omega = 1.05$ as our default.

Only ALiBi3D maintains low cross-entropy beyond the training context. Trained at $n_{ctx} = 256$, ALiBi3D maintains cross-entropy close to its in-context level out to four times that length (Figure 3.1c). The sinusoidal encoder and no positional encoding (NoPE) both degrade sharply, with cross-entropy increasing by roughly 2 nats. These findings also hold for dilated attention (introduced in Section 6.1 and discussed in Figure I.1).

4 Kernel Implementation Details

To study higher-order attention in its dense form, we need to train it in every layer at full context ($w_2 = n$). The existing Triton kernel of Roy et al. (2025) supports only a short second window ($w_2 = 32$): beyond it, its backward pass silently truncates the grid, leaving the K^2 and V^2 gradients



(a) Value-norm stability. (b) Distance exponent sweep. (c) Length extrapolation.

Figure 3.1: **Value normalization and ALiBi3D. (a)** Projection norm $\|W\|$ during training for the Q, $K^{1,2}$, and $V^{1,2}$ projections under three value-normalization recipes, in a controlled setting that isolates the value path; the inset zooms the onset of divergence. **(b)** Cross-entropy against the ALiBi3D distance exponent ω, in (blue) and out of (red) distribution; the inset zooms the optimum plateau and the dashed line marks the adopted $\omega = 1.05$. **(c)** Length extrapolation for no positional encoding (black), a sinusoidal encoder (blue), and ALiBi3D (red); the shaded region marks evaluation beyond the training length.

past the first $2B_Q$ positions of each tile at zero. At full context, the grid collapses to one tile per head, so K^2 and V^2 barely train. Enlarging the tile does not help, since it grows with $B_Q + w_2$ and lowers occupancy. We instead write a kernel that supports every window size. Because the operator is symmetric in its key–value pairs, we compute dK^1, dV^1 with one key-organized kernel, whose grid depends only on n, and dK^2, dV^2 by running it with the pairs swapped (Section F). We accumulate dQ privately in a separate query-organized kernel, which avoids atomic updates and runs up to 1.68× faster than atomic accumulation at full window ($n = 2048$, $d_{head} = 128$). The kernel also supports the ALiBi3D bias (Section 3.2) and dilation (Section 6.1), which the original has no equivalent of.

5 Scaling Laws for Higher-Order Attention

Scaling a model requires predicting what a compute budget C buys and how to allocate it between parameters N and training tokens D. Cross-entropy follows predictable power-law trends in both (Hestness et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022):

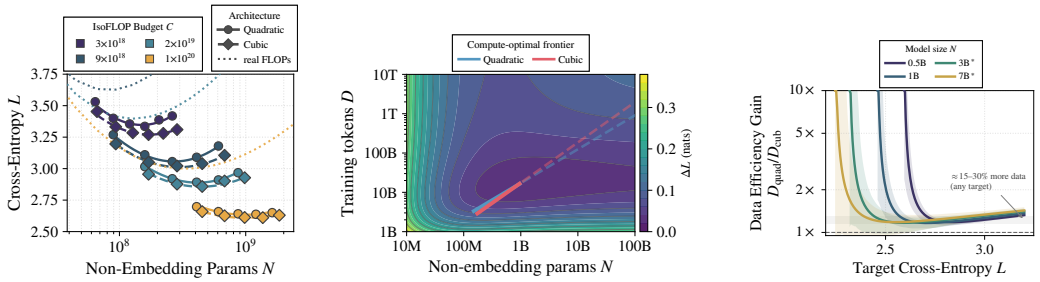
$$L(N, D) = E + \left(\frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \right)^{\gamma}, \tag{5.1}$$

where E is the irreducible loss and the outer exponent γ takes the coupled form of Busbridge et al. (2025, Eq. 1) and Videau et al. (2026). The coefficients E, A, B, α, β, and γ are fitted from training runs $\{(N_i, D_i, L_i)\}_{i=1}^{R}$, and depend on the architecture, data, and optimization recipe. We use these scaling laws to ask how the choice of attention mechanism changes the returns to model scale.

Compute-optimal models alone cannot identify these coefficients: their approximately constant token-to-parameter ratio (Hoffmann et al., 2022) traces a line through the (N, D) plane. Following Hoffmann et al. (2022), we spread runs off this line with IsoFLOP profiles (Section 5.1), which disentangles model size from training data and lets us compare the scaling behavior of higher-order and standard attention. We fit each attention mechanism independently, minimizing a Huber objective on the log-loss with multi-start L-BFGS-B, and report 95% confidence intervals from a nonparametric bootstrap; Section G gives the full procedure.

5.1 Scaling Behavior

We fit the scaling law of Equation (5.1) to IsoFLOP profiles at four compute budgets $C \in \{3 \times 10^{18}, 9 \times 10^{18}, 2 \times 10^{19}, 1 \times 10^{20}\}$. Within a budget we train models across a range of non-embedding parameter counts N; the $C = 6ND$ approximation fixes each model's token count $D = C/6N$, so larger models see proportionally fewer tokens (Figure 5.1a). This estimate undercounts cubic attention's higher per-token cost (Section K), so we plot both architectures under the shared proxy and read the compute-optimal minima under it.



(a) IsoFLOP profiles. (b) Loss advantage. (c) Token savings.

Figure 5.1: **Scaling of quadratic and cubic attention. (a)** Cross-entropy L against non-embedding parameters N at four budgets $C = 6ND$, the quadratic compute proxy, under which both architectures are plotted (Section K); quadratic (circles), cubic (diamonds). Dotted lines redraw the cubic profiles at equal real training compute ($C_{\text{real}} = 3C_{\text{fwd}}D$ at $n_{ctx} = 512$; Section 5.2). **(b)** Loss advantage $\Delta L = L_{\text{quadratic}} - L_{\text{cubic}}$ over the (N, D) plane, with each architecture's compute-optimal frontier. It stays positive throughout and grows along the frontier. **(c)** Extra tokens quadratic attention needs to match cubic attention's loss at fixed N, against target L; $N \in \{0.5, 1, 3, 7\}$ B.

Cubic attention has lower loss at the scales we measure. In every profile, the cubic curve lies below the quadratic one, reaching 0.03–0.07 nats lower cross-entropy at matched N and D, with the gap widening with data (Figure 5.1a). The fitted laws predict a positive advantage across the whole (N, D) plane (interior minimum $\Delta L = 0.024$ nats) that grows along the compute-optimal frontier (Figure 5.1b). This advantage holds at matched parameters and tokens, not matched compute: the $6ND$ proxy undercounts cubic attention's $O(n_{ctx}^2)$ per-token cost by roughly 10× at $n_{ctx} = 512$. At equal real compute (dotted curves), cubic optima shift to smaller N and higher loss, leaving quadratic attention ahead per FLOP in this budget range; however, the fitted laws predict this reverses at larger budgets (Section 5.2). (Note: For quadratic attention, the proxy closely matches real compute, so its redrawn profiles are omitted.)

The advantage lies in the irreducible loss. Higher-order attention lowers the irreducible loss by $\Delta E = E_{\text{quadratic}} - E_{\text{cubic}} = 0.134$ nats, while its data exponent β is smaller than standard attention's. A lower floor is a stronger claim than lower loss at a fixed budget: *it suggests higher-order attention reaches losses standard attention cannot, rather than merely reaching them sooner*. The sign of this gap is robust ($P(\Delta E > 0) = 0.98$ under a paired bootstrap), but its size is uncertain, since E_{cubic} is the least constrained coefficient in the fit (Table G.1). This does not conflict with the steeper parameter exponent Roy et al. (2025) report on reasoning benchmarks, which measures a different axis (Section G.3).

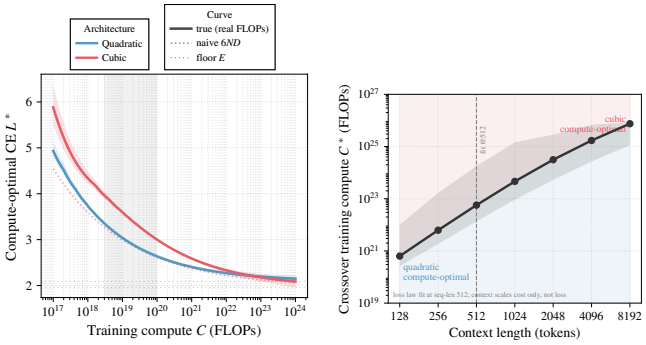
At fixed model size, quadratic training needs more tokens for a matched loss. At fixed N, quadratic attention needs 15–30% more tokens than cubic attention to match its loss, which is nearly constant across targets (Figure 5.1c). The ratio diverges as the target approaches quadratic attention's floor at that model size. The two largest sizes (3B, 7B) extrapolate the laws beyond the trained range ($N > 1.6$B).

5.2 When Is Higher-Order Attention Compute-Optimal?

Whether the additional computation is *efficient* has no single answer: as Dehghani et al. (2021) argue, efficiency claims depend on the resource, and here the verdict reverses across resources.

Per parameter and per token, higher-order attention wins. As shown in Section 5.1, it attains lower loss at matched N and D, and matches loss with fewer tokens at fixed N (Figure 5.1).

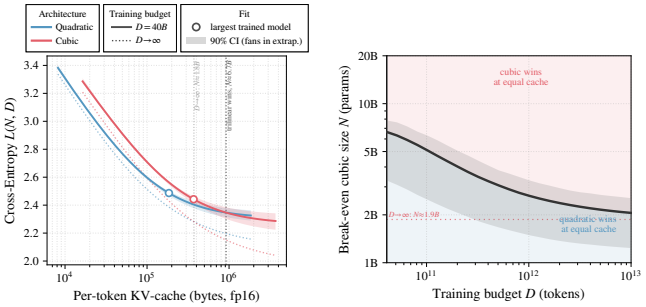
Per training FLOP, higher-order attention is predicted to win at large budgets. Under real FLOP accounting, standard attention is compute-optimal up to $C \approx 10^{23}$ at $n_{ctx} = 512$. Beyond that point, higher-order attention's lower floor outweighs its extra cost, and it overtakes standard attention (Figure 5.2a). The fits also shift the compute-optimal allocation ($N_{\text{opt}} \propto C^a$): standard attention



(a) Compute-optimal frontier.

(b) Crossover versus context.

Figure 5.2: **The efficiency verdict under compute accounting.** **(a)** Compute-optimal loss against training compute at $n_{ctx} = 512$; solid curves use true FLOP accounting, dashed the naive $6ND$ proxy, dotted lines the fitted floor E. **(b)** Crossover compute C^* against context length; shaded regions mark where each architecture is compute-optimal, the band is the 90% bootstrap interval.



(a) Equal KV-cache loss.

(b) KV-cache break-even size.

Figure 5.3: **The efficiency verdict under KV-cache accounting.** **(a)** Cross-entropy against per-token KV-cache size, at a finite training budget (solid) and in the infinite-data limit (dotted); rings mark the largest trained model, shading the confidence interval of the extrapolation. **(b)** Break-even higher-order model size against training budget D at equal KV-cache size; above the curve higher-order attention wins, below it standard attention wins.

favors parameters, while higher-order attention balances parameters and data in near-Chinchilla proportions (Hoffmann et al., 2022) (Section G).

Per byte of KV cache, higher-order attention wins only at large cache sizes. Higher-order attention caches twice per token, so at equal cache a larger standard model reaches lower loss; higher-order attention overtakes only at roughly five times the largest cache we trained (Figure 5.3a). The break-even model size falls as the training budget grows, so small budgets favor standard attention and larger budgets favor higher-order attention (Figure 5.3b).

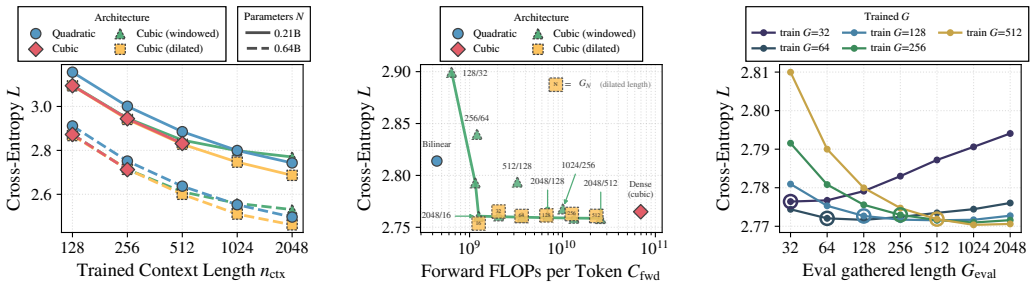
Context length determines when higher-order attention pays off. Each doubling of context raises the budget at which higher-order attention becomes compute-optimal by roughly 0.85 decades ($\approx 7\times$), because its per-token cost grows with n_{ctx} while its loss advantage does not (Figure 5.2b). This analysis assumes the scaling law fitted at $n_{ctx} = 512$ holds at other lengths with only the per-token cost rescaled, an assumption supported by the steady loss gap across trained context lengths (Figure 6.1a). These break-even budgets remain extrapolations, about $1000\times$ beyond our largest training budget, and depend on ΔE, whose sign is robust but whose magnitude is uncertain (Section 5.1). Within the budgets we train, however, higher-order attention is more efficient per parameter and per token, and standard attention per FLOP and per byte of KV cache.

6 Sparse Higher-Order Attention

Dense higher-order attention is cubic in sequence length, since the context term of Equation (K.3) scales as n_{ctx}^2 per layer and dominates C_{fwd} at moderate widths and contexts (Section K). We recover sub-cubic scaling with two sparsity patterns that preserve the higher-order inductive bias, whose per-token costs are discussed in Table K.4 and Section K.3.

6.1 Local and Multiscale Sparsity

Windowing. The windowed variant restricts each key to a local window, $j_1 \in (i - w_1, i]$ and $j_2 \in (i - w_2, i]$, so each query attends to $O(w_1 w_2)$ key pairs rather than $O(n_{ctx}^2)$ (Section 2.3). It



(a) Loss versus context length. (b) Sparsification frontier. (c) Train-versus-evaluation gather.

Figure 6.1: **Sparse variants of higher-order attention. (a)** Cross-entropy L against trained context length n_{ctx}, at two non-embedding parameter counts N; dense higher-order attention is trained only up to $n_{ctx} = 512$, beyond which its cubic cost is prohibitive. The dilated variant tracks the dense operator and holds its lead over standard attention as context grows, while the windowed variant falls behind. **(b)** Cross-entropy against forward FLOPs per token C_{fwd} at context length 2048; windowed points are labelled by w_1/w_2, dilated points by gathered length. Both sparse variants land in the dense operator's noise band at far lower per-token cost whenever one axis spans the sequence. **(c)** Cross-entropy of dilated higher-order attention trained at one gathered length and evaluated across others; rings mark train-equals-eval points. Widening the gather at evaluation helps only slightly, and only for the longest-trained models, while narrowing it hurts in every case.

matches the windowed kernel of Roy et al. (2025) and changes only the choice of w_1 and w_2, which we evaluate in Sections 6.2 and 6.3.

Dilation. A fixed window covers a shrinking share of the sequence as context grows, and stacking windowed layers extends each query's reach by at most $\max(w_1, w_2)$ per layer, so an L-layer model sees at most the last $L \max(w_1, w_2)$ tokens. To retain global reach at lower cost, we adapt LongNet's multiscale dilation (Ding et al., 2023), replacing contiguous windows with strided views that span the sequence at coarser resolutions (Section J).

6.2 Sparse Variants at Longer Context

We train standard attention, dense higher-order attention, and its windowed ($w_1 = 256$, $w_2 = 64$) and dilated (gathered length $G = 64$) variants at two model sizes, 0.21B and 0.64B parameters, and at context lengths from 128 to 2048. We train the dense operator only up to $n_{ctx} = 512$.

Sparse variants keep the advantage as context grows, if they keep full-context reach. Up to $n_{ctx} = 512$, where the dense operator is trainable, the dilated variant closely matches it (Figure 6.1a). Beyond that range, the dilated variant stays ahead of standard attention up to 2048 tokens at both model sizes. The windowed variant falls behind as context grows, because its local $w_1 \times w_2$ window covers a shrinking share of the sequence. The dilated variant avoids this because its dense K^2 axis reaches every position (Section J).

6.3 The Sparsification Frontier

Each sparse variant sets how much context it covers, and hence its compute, through the window widths w_1, w_2 for the windowed variant and the gathered length G for the dilated one. Figure 6.1b sweeps these settings at 8 blocks ($\approx$ 0.21B non-embedding parameters for standard attention) and context length 2048, plotting cross-entropy against the true per-token cost C_{fwd} (Section K) rather than the $6ND$ proxy used in Figure 5.1a.

Sparsity works when one key axis spans the full context. Both sparse variants match the dense operator's loss within noise at up to 100× lower per-token cost (Figure K.1), provided one key axis spans the whole sequence (Figure 6.1b). The dilated variant keeps its K^2 axis dense and matches dense loss at every gathered length from 16 to 512. The windowed variant matches once w_1 reaches the full context of 2048, even with w_2 as small as 16. A narrower but more expensive window does

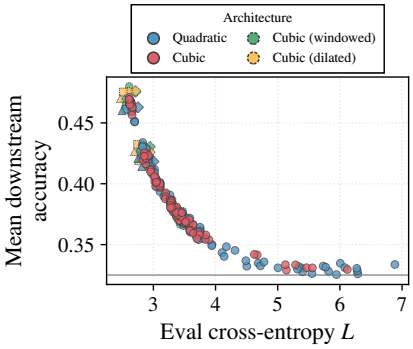


Figure 7.1: **Downstream accuracy against pretraining cross-entropy.** Mean accuracy over six benchmarks (ARC-Easy, ARC-Challenge, HellaSwag, MathQA, PIQA, WinoGrande) against eval cross-entropy (lower is better); each point is one model, for standard attention and dense, windowed, and dilated higher-order attention. All architectures fall on one shared accuracy–cross-entropy curve, so at matched cross-entropy, accuracy is the same regardless of architecture. The per-task curves are in Figure H.1.

not help, since a 512×128 window costs more per token than a 2048×16 window yet reaches a higher loss.

The gathered length can grow at evaluation only for models trained on long sequences. Increasing the gathered length G, the number of positions in each strided view, beyond its training value helps only slightly, and only for models trained on the longest sequences. Reducing G below its training value hurts in every case (Figure 6.1c).

7 Downstream Log-Likelihood Evaluation

The scaling law predicts held-out cross-entropy, but we judge an architecture on task accuracy. We therefore score the trained models on six multiple-choice benchmarks by log-likelihood, isolating whether the cross-entropy advantage of Figure 5.1a survives the move to accuracy. The benchmarks, the model set, and the per-task curves are in Section H; Figure 7.1 plots mean accuracy against pretraining cross-entropy.

All architectures lie on one shared accuracy–cross-entropy curve. On every task, accuracy is set by pretraining cross-entropy alone: at matched cross-entropy, higher-order and standard attention are indistinguishable (Figure 7.1). The cross-entropy advantage therefore reflects real capability, not an artifact of the training objective.

8 Related Work

Clift et al. (2019) introduced higher-order attention as an inductive bias for logical reasoning, and Roy et al. (2025) reported gains on reasoning, mathematics, and coding from a windowed kernel placed in every fourth layer of mixture-of-experts models. Theory gives the signal a basis: a higher-order layer can solve tasks over token triples that a standard, constant-width attention layer cannot (Sanford et al., 2023; Kozachinskiy et al., 2025). Motivated by these observations, we investigate dense language models with higher-order attention at every layer, fitting scaling laws following Hoffmann et al. (2022) and asking when the added computation is worth its cost, separating the cost axes as Dehghani et al. (2021) recommend. We discuss the wider related work in detail in Section C.

9 Conclusion

We show that higher-order attention is more efficient per parameter and per token, through a lower loss floor ($\Delta E \approx 0.134$ nats) rather than a faster data exponent, and that this gain carries over to downstream tasks. Standard attention remains more efficient per training FLOP and per byte of KV cache at the scales we train, but our fitted laws predict that higher-order attention overtakes it at larger budgets. These results position per-token computation as a third scaling axis, one worth pursuing as high-quality data runs out and compute and memory budgets grow.

AI USE STATEMENT

We used generative AI assistants throughout the preparation of this manuscript, and disclose that use in the categories of the ICLR AI Policy for Authors.

Among the tasks for which disclosure is required, we used these tools to help write proofs, to supply ingredients of those proofs, to check and sharpen the statement of mathematical claims, to help implement kernels, and to help interpret results when drafting the surrounding prose. The proofs of Sections D, E.1 and E.2 and the gradient derivation of Section F were drafted and revised with this assistance. These tools helped us prove three results whose statements are our own: that the super-linear distance penalty we adopt separates every configuration the linear penalty identifies, that the invariance group of the determinant-based three-way score contains even chunk permutations beyond the block-diagonal subgroup, and that this failure of full rotational invariance is essential rather than an artifact of the argument, since the score regains invariance under all of $SO(3)$ when the representation forms a single chunk. We used this assistance to write the kernel implementations, then verified and tested them for correctness ourselves; the design of the kernel modifications is our own. We did not use generative AI to design the experiments or to develop the conceptual framework of this work; both are the authors' own. We did not use generative AI to generate synthetic data, to clean or reformat datasets, to translate, or to perform qualitative or thematic data analysis; none of these arise in this work.

Among the tasks for which disclosure is recommended, we used these tools to draft and edit parts of the manuscript, to suggest the structure of the appendix, and to write analysis and verification code. The citations and the related-work synthesis are the authors' own.

We have reviewed all AI-assisted work. Every mathematical claim introduced or altered with this assistance was re-derived independently and checked numerically: the five gradient expressions of Section F against finite differences of the forward pass, the fan-in identities of Section K against direct enumeration, and the counterexample of Section E.1 by explicit computation. Every quantity we report is derived from logged training runs rather than produced by a model. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work is a scaling-law study of attention architectures trained on publicly available text. It does not involve human subjects, and we release no new datasets. We train on the high-quality split of Nemotron-CC (Su et al., 2025), a corpus derived from Common Crawl and released publicly under Common Crawl's terms of use. Our downstream evaluation uses standard, publicly available research benchmarks (ARC, HellaSwag, MathQA, PIQA, and WinoGrande), each used for its intended purpose of academic evaluation. The authors declare no conflicts of interest. Dense higher-order attention spends more computation per token, so at matched model size it consumes more energy per token during both training and inference than standard attention. Beyond this, we are not aware of ethical concerns other than those already inherent to training language models.

REPRODUCIBILITY STATEMENT

The higher-order attention formulation is given in Section 2.3 and its scaling-law parameterization in Section 5; Section 3 makes the operator trainable. The score normalizer is proved in Section D, and the scope of RoPE invariance for the three-way score appears in Section E.1. Full FLOPs and parameter accounting, including the conventions used to compare standard and higher-order attention, is in Section K. Source code, including the higher-order attention kernels, will be released publicly upon publication.[2] The training corpus, its preprocessing, and the full experimental setting, including the model family, optimizer, and hyperparameter sweeps, are given in Section L.

[2]Code and kernels will be released publicly upon publication.

REFERENCES

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019. URL https://arxiv.org/abs/1905.13319.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Leon Bergen, Timothy J. O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers, 2021. URL https://arxiv.org/abs/2112.00578. NeurIPS 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation scaling laws. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.08606.

Sayak Chakrabarti, Toniann Pitassi, and Josh Alman. Poly-attention: a general scheme for higher-order self-attention, 2026. URL https://arxiv.org/abs/2602.02422.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2021. URL https://arxiv.org/abs/2009.14794. ICLR 2021.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models, 2022. URL https://arxiv.org/abs/2202.01169.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

James Clift, Dmitry Doryn, Daniel Murfet, and James Wallbridge. Logic and the 2-simplicial transformer, 2019. URL https://arxiv.org/abs/1909.00668.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

Federico Danieli, Pau Rodriguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. Pararnn: Unlocking parallel training of nonlinear rnns for large language models, 2025. URL https://arxiv.org/abs/2510.21450.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Aritra Das, Dhruman Gupta, and Debayan Gupta. Linearized 2-simplicial attention, 2026. URL https://arxiv.org/abs/2608.09307.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer, 2021. URL https://arxiv.org/abs/2110.12894.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers, 2025. URL https://arxiv.org/abs/2505.01618.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023. URL https://arxiv.org/abs/2307.02486.

Alexandre Dussolle and Pietro Liò. How smoothing is n-simplicial attention?, 2025. URL https://arxiv.org/abs/2512.15600.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng

Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia,

Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023. URL https://arxiv.org/abs/2312.00752. COLM 2024.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017. URL https://arxiv.org/abs/1712.00409.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers, 2023. URL https://arxiv.org/abs/2305.19466.

Alexander Kozachinskiy, Felipe Urrutia, Hector Jimenez, Tomasz Steifer, Germán Pizarro, Matías Fuentes, Francisco Meza, Cristian B. Calderon, and Cristóbal Rojas. Strassen attention, split vc dimension and compositionality in transformers, 2025. URL https://arxiv.org/abs/2501.19215. NeurIPS 2025.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts, 2024. URL https://arxiv.org/abs/2402.07871.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, and Sebastian Jaszczur. Joint moe scaling laws: Mixture of experts can be memory efficient, 2025. URL https://arxiv.org/abs/2502.05172. ICML 2025.

Bruno Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and Marco Cuturi. Completed hyperparameter transfer across modules, width, depth, batch and duration, 2025. URL https://arxiv.org/abs/2512.22382.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models, 2023. URL https://arxiv.org/abs/2305.16264.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2409.02060.

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures, 2024. URL https://arxiv.org/abs/2403.17844. ICML 2024.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022. URL https://arxiv.org/abs/2108.12409. ICLR 2022.

Aurko Roy, Timothy Chou, Sai Surya Duvvuri, Sijia Chen, Jiecao Yu, Xiaodong Wang, Manzil Zaheer, and Rohan Anil. Fast and simplex: 2-simplicial attention in triton. *arXiv preprint arXiv:2507.02754*, 2025. URL https://arxiv.org/abs/2507.02754.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers, 2023. URL https://arxiv.org/abs/2306.02896. NeurIPS 2023.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Xuyang Shen, Dong Li, Ruitao Leng, Zhen Qin, Weigao Sun, and Yiran Zhong. Scaling laws for linear complexity language models, 2024. URL https://arxiv.org/abs/2406.16690. EMNLP 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://arxiv.org/abs/2412.02595.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Mathurin Videau, Badr Youbi-Idrissi, David Lopez-Paz, and Kartik Ahuja. Skaling: Chinchilla's exponents meet kaplan's coupling, 2026. URL https://arxiv.org/abs/2608.07222.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? limits of llm scaling based on human-generated data, 2024. URL https://arxiv.org/abs/2211.04325.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL https://arxiv.org/abs/2201.11903.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2203.03466.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2020. URL https://arxiv.org/abs/2007.14062.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Appendices

A Notation **18**

B Limitations **18**

C Expanded Related Work **20**

D The Score Normalizer for Higher-Order Attention **21**

E Positional Encoding: Rotational Invariance and Degeneracy **22**

E.1 Scope of Rotational Invariance in the Determinant-Based Three-Way Form 22

E.2 Degeneracy of the Linear Extension 24

F The Backward Pass of Higher-Order Attention **25**

F.1 Forward Pass and the Original Coverage Failure 25

F.2 What the Forward Pass Leaves Behind. 25

F.3 Gradient of the Attention Weights 25

F.4 Gradients of the Value Projections 25

F.5 Gradient of the Scores 26

F.6 Gradients of the Query and Key Projections 26

F.7 Why Three Kernels Suffice. 27

G Additional Results: Fitted Scaling Laws **27**

G.1 Coefficient Estimation 27

G.2 Fitted Coefficients 28

G.3 Comparison with the Parameter-Exponent Fit of Roy et al. 28

H Additional Results: Downstream Evaluation **29**

I Additional Results: Positional Encoding under Dilated Attention **29**

J Dilated Higher-Order Attention: Construction and Cost **30**

K FLOP and Parameter Comparison: Standard vs. Higher-Order Attention **31**

K.1 Theoretical FLOP Comparison. 31

K.2 Parameter Accounting 32

K.3 Sparse Variants 33

K.4 Summary 34

L Experimental Setting **35**

A NOTATION

Table A.1 lists the mathematical symbols used throughout the paper, grouped by role: dimensions and counts, indices, language modeling, attention, sparse attention, positional encoding, and scaling laws. Bold denotes vectors, matrices, and tensors; plain denotes scalars; subscripts are component indices. Acronyms are expanded on first use and in the glossary.

We state a few conventions the per-row entries cannot show in full. A handful of letters recur across roles, most notably A (attention weights and the parameter-law coefficient) and Z (a projection label in $\boldsymbol{W}^Z$ and the softmax partition function $Z(\boldsymbol{z})$); the enclosing group and the definition column disambiguate each use. The parameter symbols play distinct roles: N is the whole-model non-embedding count that serves as the scaling-law variable, whereas P is the exact per-block learnable count from which N is assembled; because the embeddings are untied, the output projection is a genuine weight matrix and is counted in N (Section K.2).

Table A.1: Symbols grouped by role.

Symbol	Meaning
Dimensions and counts	
n_{ctx}	Sequence (context) length
n_{vocab}	Vocabulary size
d_{model}	Residual-stream width
d_{head}	Per-head width, $d_{model} = n_{head} d_{head}$
n_{head}	Number of attention heads
n_{layer}	Number of transformer layers
d_{ffn}	Multi-Layer Perceptron (MLP) hidden width, $d_{ffn} = \lfloor \rho_{ffn}\, d_{model} \rfloor$
ρ_{ffn}	Feed-Forward Network (FFN) width ratio (2.75)
n_{ffn}	Number of MLP weight matrices
Indices	
i; j_1, j_2	Query position; key positions
a, b	Channel indices within a head, $a, b \in \{1, \ldots, d_{head}\}$
Language modeling	
$\mathcal{V}$	Vocabulary, $\mathcal{V} = \{1, \ldots, n_{vocab}\}$
$\boldsymbol{x}^{(<i)}$	Context tokens preceding position i
$f, \boldsymbol{\theta}$	Network mapping context to logits, and its parameters
$\boldsymbol{z}^{(i)}$	Logits at position i, $\boldsymbol{z}^{(i)} \in \mathbb{R}^{n_{vocab}}$
$\sigma, Z(\boldsymbol{z})$	Softmax and its partition function
$\mathcal{L}, \lambda_Z$	Per-token loss (cross-entropy plus z-loss) and z-loss weight (10^{-4})
Attention	
$\boldsymbol{X}$	Head input, $\boldsymbol{X} \in \mathbb{R}^{n_{ctx} \times d_{model}}$
$\boldsymbol{W}^Z$	Projection, $Z \in \{Q, K^1, K^2, V^1, V^2\}$ (standard: $\{Q, K, V\}$; the label Z differs from $Z(\boldsymbol{z})$), $\boldsymbol{W}^Z \in \mathbb{R}^{d_{model} \times d_{head}}$
$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$	Query, key, value projections, each $\in \mathbb{R}^{n_{ctx} \times d_{head}}$
$\boldsymbol{S}, \mathbf{S}$	Score matrix $\in \mathbb{R}^{n_{ctx} \times n_{ctx}}$ (quadratic) / tensor $\in \mathbb{R}^{n_{ctx} \times n_{ctx} \times n_{ctx}}$ (higher-order)
A, O	Attention weights after softmax ($A_{i,j}$; A_{i,j_1,j_2}), and single-head output $O_{i,a}$
Sparse attention	
w_1, w_2	Window widths of the windowed variant; each query attends to $O(w_1 w_2)$ key pairs (Section 6.1)
G	Gathered length of the dilated variant: the segment size on the gathered (K^1) axis, the K^2 axis staying dense (Section 6.1)
Positional encoding	
ℓ, p	Chunk index and number of 3-channel chunks, $d_{head} = 3p$
$\boldsymbol{R}$, $O(n)$, $SO(n)$	Orthogonal matrix ($\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}$) on $\mathbb{R}^n$; orthogonal group; rotation subgroup (det $\boldsymbol{R} = 1$)
Scaling laws	
C	Compute budget
N	Non-embedding parameter count
P	Exact per-block learnable parameter count
D	Training tokens
L, E	Cross-entropy loss and irreducible loss
A, B	Parameter- and data-law coefficients in $L(N, D)$
α, β, γ	Scaling-law exponents
R	Number of fitted training runs

B LIMITATIONS

We gather here the caveats that qualify the results, in rough order of how much they bear on the headline claim. The first five concern the fitted laws and the crossovers drawn from them; the next

two concern the scope of the study; the last three concern the experiments, the fitting procedure, and the training recipe. We cannot be absolutely certain of any of these, and for each we note what future study would resolve.

We cannot be absolutely certain of the size of the floor difference. The advantage of higher-order attention rests on a lower irreducible loss, $\Delta E = 0.134$ nats, and this is the least-pinned coefficient of the fit. Its direction is robust ($P(\Delta E > 0) = 0.98$ under a paired bootstrap), but its size is not: E for higher-order attention carries the widest confidence interval of any coefficient, pinned mainly by the endpoints of the compute range we train (Table G.1). Firming it would need a compute budget above 10^{20}, a longer lever arm than any profile here provides; a simulated intermediate budget narrows the interval only slightly. Every crossover verdict that rests on ΔE inherits this uncertainty.

The resource crossovers are extrapolations, not measurements. Higher-order attention becomes compute-optimal per training FLOP only past $C \approx 10^{23}$ at $n_{ctx} = 512$ (Figure 5.2a), and overtakes standard attention per byte of Key-Value (KV) cache only at roughly five times the largest cache we train (Figure 5.3b; Section 5.2). The compute crossover sits about three decades beyond the largest budget we reach; the KV-cache crossover sits closer. We therefore report them as extrapolations of the fitted laws rather than as measured verdicts; closing the gap needs runs at those budgets, which we do not have.

The context-scaling of the crossover assumes the law transfers across sequence lengths. We fit the scaling law at $n_{ctx} = 512$ and, when we ask how the crossover moves with context, rescale only the per-token cost, holding the achieved loss fixed (Section 5.2). This treats the floor and exponents as properties of the architecture and data rather than of the training context length. The assumption is supported here: the loss gap between standard and higher-order attention holds steady across the context lengths we train (Figure 6.1a), so within that range the law moves with context mainly through cost. We cannot be certain it holds at the longer contexts where the crossover figure extrapolates (Figure 5.2b), and a fit at a second context length would test it directly.

The token savings at the largest model sizes extrapolate beyond the trained range. The claim that standard attention needs 15–30% more tokens to match higher-order attention's loss holds at fixed model size across our targets, but the two largest of these, 3B and 7B, lie beyond the parameter range we train ($N > 1.6$B; Section 5.1). At those sizes the ratio is a projection of the fitted laws, not a measurement, and it shares the floor uncertainty of the fit. Training at billion-parameter scale would replace the projection with a direct comparison.

In the regime we train, standard attention is the better spend per FLOP and per cached byte. Higher-order attention wins per parameter and per token (Figures 5.1b and 5.1c), but the resources that bind training and inference budgets today are training FLOPs and KV cache, and on both standard attention is compute-optimal across the range we reach (Figures 5.2a and 5.3a; Section 5.2). Standard attention's per-cache advantage is the firmest of these: higher-order attention caches twice per token, so at equal cache a slightly larger standard model reaches lower loss (Figure 5.3a), a fact of cache width that does not depend on the fitted floor. Higher-order attention is thus a training-compute-optimal architecture in prospect, not an inference-optimal one today.

Every run uses one corpus and one tokenizer. We fit both scaling laws on a single data recipe (Section L), and the coefficients, including the floor difference, depend on the architecture, data, and optimization recipe (Section 5). Whether the floor advantage of higher-order attention holds under a different corpus, tokenizer, or data mixture is a question our data cannot answer. Repeating the fit on a second recipe would show how much of the advantage is recipe-specific.

Models are scaled along a single width-to-depth curve. We vary model size by moving along one aspect-ratio schedule, $d_{model} = 128\, n_{layer}$, so width and depth grow together in fixed proportion (Table L.1). The scaling laws therefore describe how higher-order attention behaves along that curve, not how it trades off against shape at fixed size. A three-way score couples positions across the sequence, and whether its advantage depends on having enough heads or depth to exploit that coupling is something a shape sweep, not this study, would settle.

The sparse variants are validated only to moderate context. The dilated variant tracks the dense operator and holds its lead over standard attention out to 2048 tokens (Figures 6.1a and 6.1b; Section 6.2), the longest context we train, and its dilation schedule is adapted from LongNet (Ding et al., 2023) rather than derived for this operator. Whether the dilated variant keeps full-context reach and its loss advantage at the much longer contexts where sparsity matters most is untested here. Confirming it would need training at those lengths and, ideally, a dilation schedule tuned to the three-way score rather than borrowed.

Each law is fit from few points, most of them single-seed. We fit six coefficients per architecture from 21 points, far fewer than the hundreds behind established scaling laws (Hoffmann et al., 2022), and most (N, D) cells are trained at a single seed (Section L). The fits are tight where we measure (mean relative error $\approx 0.2\%$; Table G.1), but the coefficients, and the floor difference in particular, are constrained by sparse coverage rather than by repeated observation. More points, and replicate seeds at the cells that most move the floor, would tighten the fit and test its stability.

We assume the learning rate transfers across model sizes without verifying it per size. Each architecture's learning rate is tuned at the base size and transferred to every larger size with CompleteP (Dey et al., 2025; Mlodozeniec et al., 2025), rather than retuned at each scale (Section L). If transfer holds imperfectly at the larger sizes, some of the fitted loss, and any difference between the two architectures in how well it holds, would reflect the learning rate rather than the attention mechanism. A per-size learning-rate sweep would separate the two. We verify muP-style *width* transfer for higher-order attention directly with a coordinate check (Figure L.1), and rely on CompleteP (Dey et al., 2025) for *depth* transfer, which it is designed to provide.

C Expanded Related Work

Higher-order attention. Clift et al. (2019) introduced attention based on a three-way score between a query and two keys. Similar triangular mechanisms are used to update pair representations in Edge Transformers (Bergen et al., 2021) and AlphaFold (Jumper et al., 2021). Theoretically, higher-order variants can solve tasks over token triples that a standard, constant-width attention layer cannot (Sanford et al., 2023; Kozachinskiy et al., 2025; Chakrabarti et al., 2026). In the context of language modeling, Roy et al. (2025) proposed a windowed kernel placed in every fourth layer of mixture-of-experts models. They observed a larger scaling exponent on reasoning and knowledge benchmarks when fitting a power law to active parameters at a fixed token budget, while Das et al. (2026) later introduced a linear-time approximation. However, whether higher-order attention at *every* layer of a dense model follows a joint scaling law $L(N, D)$ with coefficients distinct from standard attention has remained open. We establish this law for both higher-order and standard attention, demonstrating that their coefficients do indeed separate (Table G.1).

Lower-order attention. Conversely, a substantial body of research seeks to reduce token interaction costs below the standard pairwise level. Examples include linear attention (Katharopoulos et al., 2020), random-feature approximations (Choromanski et al., 2021), and state-space models (Gu & Dao, 2023; Dao & Gu, 2024). Shen et al. (2024) fitted scaling laws to several of these architectures and found scaling capacity comparable to softmax attention, while Poli et al. (2024) reported that hybrid designs match or improve on the Transformer++ compute-optimal frontier. While these studies evaluate what performance is retained when per-token computation decreases, the potential data-efficiency gains of *increasing* this computation are less explored. We quantify this upside, showing that at a fixed parameter count, the higher-order operator matches baseline loss while requiring 15–30% fewer training tokens (Figure 5.1).

Scaling laws. It is well established that model loss follows a power law with respect to parameters, data, and compute (Hestness et al., 2017; Kaplan et al., 2020). We adopt the joint form $L(N, D)$ introduced by Hoffmann et al. (2022), which is fitted from IsoFLOP profiles. Subsequent work has fitted similar laws to evaluate interventions by comparing coefficients, such as routing (Clark et al., 2022; Krajewski et al., 2024; Ludziejewski et al., 2025), data repetition (Muennighoff et al., 2023), and distillation (Busbridge et al., 2025). While mixture-of-experts models (Dai et al., 2024; Muennighoff et al., 2025) scale parameters while keeping per-token computation fixed, higher-order attention does the reverse: it increases per-token compute at a fixed parameter count. This axis

of scaling is particularly relevant as the stock of public human-generated text becomes a binding constraint (Villalobos et al., 2024). Following Dehghani et al. (2021), we separate these cost axes. Our fits demonstrate that increasing per-token computation effectively lowers both the irreducible loss and the overall token requirement (Section 5.1).

Efficient and sparse attention. Architectures like Sparse Transformers (Child et al., 2019), Longformer (Beltagy et al., 2020), and BigBird (Zaheer et al., 2020) rely on sparse patterns to reduce standard attention from quadratic to subquadratic complexity. Similarly, LongNet (Ding et al., 2023) introduced the dilated pattern that inspires our own dilated variant. However, the computational trade-offs in our setting are fundamentally different. The dense higher-order operator scales cubically with sequence length; by striding one leg, we reduce this to quadratic complexity, matching dense standard attention. We find that introducing this specific sparsity structure preserves the data-efficiency benefits of the dense operator, whereas a windowed variant with fixed local windows degrades as context length increases (Figure 6.1 and Section 6.2).

Positional encoding. Rotary Position Embedding (RoPE) (Su et al., 2023) encodes relative position via rotations that preserve the invariance of the pairwise dot product. Roy et al. (2025) noted that a standard three-way score loses this property, restoring it with a determinant-based score that Dussolle & Liò (2025) later generalized to higher arities. An alternative approach to positional encoding is Attention with Linear Biases (ALiBi) (Press et al., 2022), which applies an additive distance penalty to enable extrapolation beyond the training context length. We show that applying a similar additive penalty across the three pairwise distances successfully allows the unmodified three-way score to extrapolate. Our ALiBi3D implementation (Section 3.2) maintains stable loss at up to four times the original training length (Figure 3.1c).

D The Score Normalizer for Higher-Order Attention

The $\sqrt{d_{head}}$ in Equation (2.6) holds the pre-softmax score at unit variance at initialization. Here we derive it, and show that the same normalizer holds at any attention order.

Write the three-way score of Equation (2.6) with a normalizer Z_S to be determined,

$$S_{i,j_1,j_2} = \frac{1}{Z_S} \sum_{a=1}^{d_{head}} Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}, \tag{D.1}$$

and suppose the entries $Q_{i,a}$, $K^1_{j_1,a}$ and $K^2_{j_2,a}$ are mutually independent, both across the three tensors and across coordinates a, each with mean 0 and variance 1.

Proposition 1 (Cubic score normalizer)**.** *Under these assumptions the choice* $Z_S = \sqrt{d_{head}}$ *makes* $\mathbb{E}[S_{i,j_1,j_2}] = 0$ *and* $\mathrm{Var}[S_{i,j_1,j_2}] = 1$.

Proof. A sum of d_{head} independent, mean-0, variance-1 terms has mean 0 and variance d_{head}, since independence makes the cross-terms vanish and the variances add. It remains to check that each summand $Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}$ is such a term.

Each summand is a product of independent factors, so both its mean and its variance factorize across the three tensors. The mean vanishes,

$$\mathbb{E}[Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}] = \mathbb{E}[Q_{i,a}]\, \mathbb{E}[K^1_{j_1,a}]\, \mathbb{E}[K^2_{j_2,a}] = 0, \tag{D.2}$$

and, being mean 0, its variance equals its second moment,

$$\mathrm{Var}[Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}] = \mathbb{E}[Q^2_{i,a}]\, \mathbb{E}[(K^1_{j_1,a})^2]\, \mathbb{E}[(K^2_{j_2,a})^2] = 1. \tag{D.3}$$

The d_{head} summands are independent across a, so the unnormalized contraction has mean 0 and variance d_{head},

$$\mathbb{E}\left[\sum_{a=1}^{d_{head}} Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}\right] = 0, \qquad \mathrm{Var}\left[\sum_{a=1}^{d_{head}} Q_{i,a} K^1_{j_1,a} K^2_{j_2,a}\right] = d_{head}. \tag{D.4}$$

The normalizer rescales this to $\mathrm{Var}[S_{i,j_1,j_2}] = d_{head}/Z_S^2$, which equals 1 precisely when $Z_S = \sqrt{d_{head}}$, recovering Equation (2.6). □

Proposition 2 (Any-order score normalizer)**.** *The normalizer $\sqrt{d_{head}}$ is independent of the number of factors in the contraction, depending only on the summation dimension d_{head}.*

Proof. Nothing in the proof of Proposition 1 used that there were three factors. For an m-linear score $\sum_{a=1}^{d_{head}} \prod_{r=1}^{m} X_{r,a}$ with the $X_{r,a}$ mutually independent, mean 0, variance 1, each summand still has mean 0 and variance 1,

$$\mathbb{E}\left[\prod_{r=1}^{m} X_{r,a}\right] = \prod_{r=1}^{m} \mathbb{E}[X_{r,a}] = 0, \qquad \mathrm{Var}\left[\prod_{r=1}^{m} X_{r,a}\right] = \prod_{r=1}^{m} \mathbb{E}[X_{r,a}^2] = 1. \tag{D.5}$$

The d_{head} summands are independent across a, so the contraction has variance d_{head},

$$\mathrm{Var}\left[\sum_{a=1}^{d_{head}} \prod_{r=1}^{m} X_{r,a}\right] = d_{head}, \tag{D.6}$$

and the normalizer remains $\sqrt{d_{head}}$ for every m. □

Beyond initialization. The $\sqrt{d_{head}}$ normalizer relies on the coordinate summands being uncorrelated across a, which holds at initialization but need not survive training. As the query and keys align, the summands correlate, the $d_{head}(d_{head}-1)$ off-diagonal terms of the variance stop cancelling, and the normalizer grows with the correlation. This is the higher-order analogue of the shift from $1/\sqrt{d_{head}}$ to $1/d_{head}$ attention that maximal update parametrization prescribes once query and key become correlated (Yang et al., 2021). We record the interpolation below for a reader who wishes to renormalize in a correlated regime.

Proposition 3 (Cubic score normalizer under correlated coordinates)**.** *Suppose the summands $Y_a = Q_{i,a}K^1_{j_1,a}K^2_{j_2,a}$ have unit variance and a common pairwise correlation $\rho = \mathrm{Corr}[Y_a, Y_b]$ for $a \neq b$. Then the choice*

$$Z_S = \sqrt{d_{head}\,[1 + (d_{head} - 1)\rho]} \tag{D.7}$$

makes $\mathrm{Var}[S_{i,j_1,j_2}] = 1$.

Proof. The unnormalized contraction $\sum_{a=1}^{d_{head}} Y_a$ has variance

$$\mathrm{Var}\left[\sum_{a=1}^{d_{head}} Y_a\right] = \sum_{a=1}^{d_{head}} \mathrm{Var}[Y_a] + \sum_{a\neq b} \mathrm{Cov}[Y_a, Y_b] = d_{head} + d_{head}(d_{head}-1)\rho, \tag{D.8}$$

since each of the $d_{head}(d_{head}-1)$ off-diagonal terms equals ρ. Dividing by $Z_S^2 = d_{head}[1+(d_{head}-1)\rho]$ gives unit variance. □

Equation (D.7) interpolates the two regimes: at $\rho = 0$ it recovers $Z_S = \sqrt{d_{head}}$ of Proposition 1, and at $\rho = 1$ it gives $Z_S = d_{head}$, the linear normalizer. Any fixed $\rho > 0$ makes $Z_S \sim d_{head}\sqrt{\rho}$ at large d_{head}, so the square-root scaling is a knife-edge that holds only while $\rho \lesssim 1/d_{head}$.

E Positional Encoding: Rotational Invariance and Degeneracy

E.1 Scope of Rotational Invariance in the Determinant-Based Three-Way Form

In quadratic attention the score is a dot product, $S_{i,j} \propto \langle \boldsymbol{Q}_i, \boldsymbol{K}_j \rangle$ (Equation (2.4)), which is invariant under any orthogonal $\boldsymbol{R}$ applied jointly to both arguments, $\langle \boldsymbol{R}\boldsymbol{Q}_i, \boldsymbol{R}\boldsymbol{K}_j \rangle = \langle \boldsymbol{Q}_i, \boldsymbol{K}_j \rangle$. RoPE uses this invariance: a shared per-position rotation cancels to leave a function of relative position. This holds for every $\boldsymbol{R} \in SO(d_{head})$, so the invariance is basis-independent: it does not depend on how the channels are grouped. To extend RoPE to three-way interactions, Roy et al. (2025) replace the dot product with a 3D determinant, which is likewise invariant under a proper rotation applied jointly to its three arguments, $\det(\boldsymbol{R}\boldsymbol{a}, \boldsymbol{R}\boldsymbol{b}, \boldsymbol{R}\boldsymbol{c}) = \det(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ for $\boldsymbol{R} \in SO(3)$. As we show, this three-way invariance is weaker: it holds only within each 3-channel chunk, not under general rotations of the full channel space, so unlike the quadratic dot product it depends on the chosen grouping and loses RoPE's basis-independence. Throughout, $O(n)$ denotes the orthogonal transformations of $\mathbb{R}^n$, the

matrices $\boldsymbol{R}$ with $\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}$, and $SO(n)$ the rotations among them, those with $\det \boldsymbol{R} = 1$; a reflection flips the sign of the determinant, so the relevant group is $SO(3)$ rather than $O(3)$.

For clarity we take the head dimension to be $d_{head} = 3p$, so the channel index $a \in \{1, \dots, d_{head}\}$ partitions into p consecutive *chunks* of three channels, the ℓ-th chunk being channels $\{3\ell - 2,\ 3\ell - 1,\ 3\ell\}$. We write $\boldsymbol{e}_a \in \mathbb{R}^{d_{head}}$ for the standard basis vector along channel a, so $\boldsymbol{e}_a$ lies in chunk $\lceil a/3 \rceil$. Write $Q_i^{(\ell)} = (Q_{i,3\ell-2},\ Q_{i,3\ell-1},\ Q_{i,3\ell}) \in \mathbb{R}^3$ for the ℓ-th chunk of the query at token i, and $K_{j_1}^{1,(\ell)}, K_{j_2}^{2,(\ell)} \in \mathbb{R}^3$ for the corresponding chunks of the two keys. The naive rotary extension of the score S_{i,j_1,j_2} (Equation (2.6)) rotates the query and both keys and contracts them (Equation (3.4)), but the three rotations do not pair off and the score keeps an absolute-position dependence. Roy et al. (2025) instead build the logit from a sum of 3×3 determinants over the chunks, whose joint-rotation invariance plays the role that the invariance of the dot product plays for RoPE; we write this Fast and Simplex (FS) determinant score as

$$S^{\mathrm{FS}}_{i,j_1,j_2} = \sum_{\ell=1}^{p} \det\Big(Q_i^{(\ell)},\ K_{j_1}^{1,(\ell)},\ K_{j_2}^{2,(\ell)}\Big), \tag{E.1}$$

in place of the contraction S_{i,j_1,j_2}, with a joint softmax over (j_1, j_2) and the value sum of Equation (2.7) unchanged; each chunk is a column of its determinant. The construction also admits a head dimension that is not a multiple of 3, with the leftover channels serving a separate role.

Proposition 4 (Rotation invariance is chunk-local)**.** *For $d_{head} = 3p$ with $p \geq 2$, S^{FS} is not invariant under a general rotation $\boldsymbol{R} \in SO(d_{head})$ of the channels shared by the query and both keys.*

Proof. Apply a rotation $\boldsymbol{R} \in SO(d_{head})$ to the channels of the query and both keys jointly, as RoPE applies its position rotation to query and key alike. A block-diagonal $\boldsymbol{R}$ preserves every summand of Equation (E.1); a rotation that mixes channels across a chunk boundary need not, and one such rotation settles the claim. For the block-diagonal subgroup, take $\boldsymbol{R} = \mathrm{diag}(\boldsymbol{R}_1, \dots, \boldsymbol{R}_p)$ with each $\boldsymbol{R}_\ell \in SO(3)$ acting within the ℓ-th chunk. Multiplicativity of the determinant leaves each summand fixed,

$$\begin{aligned} \det\Big(\boldsymbol{R}_\ell Q_i^{(\ell)}, \boldsymbol{R}_\ell K_{j_1}^{1,(\ell)}, \boldsymbol{R}_\ell K_{j_2}^{2,(\ell)}\Big) &= \det(\boldsymbol{R}_\ell)\, \det\Big(Q_i^{(\ell)}, K_{j_1}^{1,(\ell)}, K_{j_2}^{2,(\ell)}\Big) \\ &= \det\Big(Q_i^{(\ell)}, K_{j_1}^{1,(\ell)}, K_{j_2}^{2,(\ell)}\Big), \end{aligned} \tag{E.2}$$

since $\det \boldsymbol{R}_\ell = 1$. A rotation that mixes channels across chunks has no such guarantee. Each summand is a 3×3 determinant whose columns are the query and two keys restricted to that chunk, so it is nonzero exactly when those three columns are linearly independent and vanishes as soon as any column is zero. A single rotation that empties a column settles the proposition. Take $d_{head} = 6$ (so $p = 2$) and set the query and keys to the first three basis vectors, $Q = \boldsymbol{e}_1$, $K^1 = \boldsymbol{e}_2$, $K^2 = \boldsymbol{e}_3$, so all three fill chunk 1 and chunk 2 is empty:

$$S^{\mathrm{FS}} = \underbrace{\det \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{chunk 1, independent}} + \underbrace{\det(\boldsymbol{0})}_{\text{chunk 2}} = 1. \tag{E.3}$$

Let $\boldsymbol{U} \in SO(6)$ rotate the $(3, 4)$ plane by $\pi/2$, sending $\boldsymbol{e}_3 \mapsto \boldsymbol{e}_4$ and $\boldsymbol{e}_4 \mapsto -\boldsymbol{e}_3$. It carries $K^2 = \boldsymbol{e}_3$ to $\boldsymbol{U}K^2 = \boldsymbol{e}_4$, moving that column out of chunk 1 and into chunk 2, so each chunk now holds a zero column and both determinants vanish:

$$S^{\mathrm{FS}}_{\boldsymbol{U}} = \underbrace{\det \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{chunk 1, zero column}} + \underbrace{\det \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{chunk 2, zero columns}} = 0 \neq 1 = S^{\mathrm{FS}}. \tag{E.4}$$

Since $\boldsymbol{U}$ is a genuine rotation, $\det \boldsymbol{U} = 1$, this shows S^{FS} is not invariant under general $\boldsymbol{R} \in SO(d_{head})$. □

The failure is intrinsic to the chunking, not an artifact of the argument: at $p = 1$ the score is the single determinant $\det(Q_i^{(1)}, K_{j_1}^{1,(1)}, K_{j_2}^{2,(1)})$, invariant under all of $SO(3)$ by Equation (E.2), and it breaks for every $p \geq 2$, once the channels split across chunks and a rotation can straddle a boundary. Because the surviving invariance depends on the chosen chunking rather than on a fixed coordinate frame, we encode position with ALiBi3D instead (Section 3.2), an additive distance bias that needs no preferred decomposition.

E.2 Degeneracy of the Linear Extension

Here we show that the natural linear extension of ALiBi to triples is degenerate—it collapses an entire family of distinct configurations onto a single penalty—and that the super-linear exponent $\omega > 1$ of Equation (3.6) removes exactly this family, because translation invariance alone leaves the exponent undetermined. ALiBi adds a distance-dependent bias to the attention logits with two properties: it is translation invariant, and it prefers nearby tokens monotonically. Recall the ALiBi bias of Section 3.2, restated for convenience,

$$b(i + t, j + t) = b(i, j), \qquad b(i, j) = -m|i - j|. \tag{E.5}$$

A natural extension to the higher-order setting penalizes the total pairwise distance linearly,

$$b(i, j_1, j_2) \;=\; -m\big(|i - j_1| + |i - j_2|\big). \tag{E.6}$$

Each term depends only on a difference $i - j_k$, so this form satisfies the translation requirement of Equation (3.5) exactly as ALiBi3D does. What separates the two forms is degeneracy.

Equation (E.6) is symmetric in j_1 and j_2 and encourages all three indices to remain close, but it reads the two distances only through their sum, so distinct structural configurations of (i, j_1, j_2) receive identical penalties. Writing causal triples, with $j_1, j_2 \leq i$ as in Section K,

$$b(6, 3, 3) \;=\; -m \cdot (3 + 3) \;=\; -6m, \tag{E.7}$$

while

$$b(6, 6, 0) \;=\; -m \cdot (0 + 6) \;=\; -6m. \tag{E.8}$$

The first configuration is a query attending to two keys equidistant behind it, the second a query attending to itself and to one distant key: qualitatively different structures that Equation (E.6) penalizes identically.

The super-linear penalty of Equation (3.6), restated for convenience,

$$b(i, j_1, j_2) \;=\; -m\left(|i - j_1|^{\omega} + |i - j_2|^{\omega}\right), \tag{E.9}$$

removes this collapse once $\omega > 1$. Returning to the example above,

$$b(6, 3, 3) = -m \cdot (3^{\omega} + 3^{\omega}) = -2m \cdot 3^{\omega}, \tag{E.10}$$

$$b(6, 6, 0) = -m \cdot (0^{\omega} + 6^{\omega}) = -m \cdot 6^{\omega}, \tag{E.11}$$

and $2 \cdot 3^{\omega} = 6^{\omega}$ rearranges to $2 = 2^{\omega}$, which holds only at $\omega = 1$.

The separation is general rather than particular to this example. The linear form assigns two configurations the same penalty exactly when their distances share a sum, so fix that sum Σ and write the super-linear penalty of a pair as

$$g(r) = r^{\omega} + (\Sigma - r)^{\omega}, \qquad r = \min(|i - j_1|, |i - j_2|) \in [0, \Sigma/2]. \tag{E.12}$$

For $\omega > 1$ the map $r \mapsto r^{\omega}$ is strictly convex, so g is strictly convex on $[0, \Sigma]$ and symmetric about $\Sigma/2$. Symmetry places its unique stationary point at $r = \Sigma/2$, and strict convexity makes that point the unique minimizer; a strictly convex function is therefore strictly decreasing on either side of its minimizer, hence strictly decreasing on $[0, \Sigma/2]$. Distinct pairs sharing a sum have distinct values of r, so they receive distinct penalties, and the exponent separates every collision the linear form creates.

The exponent removes this family of collisions without making the bias injective: pairs whose distances have different sums can still coincide, and at $\omega = 2$ the pairs $(3, 4)$ and $(0, 5)$ both give 25. The difference is one of kind: the linear form collapses an entire family at once, every pair sharing a sum, whereas what survives are isolated coincidences between particular distances. A concave exponent $\omega < 1$ would separate the same family, by the same argument with g strictly increasing on $[0, \Sigma/2]$, so degeneracy alone does not select a super-linear penalty. We take $\omega > 1$ because it accentuates large separations rather than compressing them, and the optimum sits just above 1 (Figure 3.1b).

F The Backward Pass of Higher-Order Attention

This section derives the five gradients computed by the kernel of Section 4, in the notation of Section 2.3. Throughout, dX denotes the gradient of the training loss with respect to a tensor X, and dO is given by the layer above. Sums over j_1 and j_2 run over the windows w_1 and w_2; at full attention they run over all causal positions.

F.1 Forward Pass and the Original Coverage Failure

For a sequence of n tokens, query i attends to key pairs (j_1, j_2) within the windows w_1 and w_2 (Sections 2.3 and 6.1), giving up to nw_1w_2 scores per head. Following FlashAttention (Dao et al., 2022), the forward pass never materializes this score tensor: each Graphics Processing Unit (GPU) thread block owns a tile of B_Q queries, loops over K^1 positions, streams blocks of K^2 and V^2 on-chip, adds the ALiBi3D bias, and maintains a running softmax. Only the output and each query's log-sum-exp M_i are written back.

The original design of Roy et al. (2025) runs the backward pass as two kernels. The first, organized around blocks of K^1 positions, accumulates dK^1 and dV^1 privately. The second, around tiles of $B_Q + w_2$ positions, is launched twice and accumulates dQ, dK^2, and dV^2 for B_Q queries per tile. Together they cover every query when $w_2 \le B_Q$, but when $w_2 > B_Q$ they reach only the first $2B_Q$ queries of each tile, leaving the rest without dK^2 or dV^2. At full context ($w_2 = n$) the grid holds a single tile per head, so dK^2 and dV^2 receive almost no gradient and these projections do not train, with no runtime error. Enlarging B_Q does not fix this: the second kernel's on-chip tile of K^2 and V^2 grows with $B_Q + w_2$, and larger tiles lower occupancy, so covering $w_2 = n$ demands a tile as long as the sequence, one block per head. The derivation below shows why our three key- and query-organized kernels cover every window size at any w_2.

F.2 What the Forward Pass Leaves Behind

The forward pass writes only the output O and one scalar per query, the log-sum-exp of that query's scores,

$$M_i = \log \sum_{j_1, j_2} \exp\left(S_{i,j_1,j_2}\right). \tag{F.1}$$

The attention weights of Equation (2.7) are recovered from it by

$$A_{i,j_1,j_2} = \exp\left(S_{i,j_1,j_2} - M_i\right). \tag{F.2}$$

A backward kernel therefore recomputes a score from Equation (2.6), subtracts M_i, and exponentiates, rather than reading a stored attention weight. This is what keeps the third-order tensors $\mathbf{S}$ and $\mathbf{A}$ out of memory.

F.3 Gradient of the Attention Weights

Each output row is a weighted sum of the pair-indexed values, so the gradient with respect to one attention weight contracts dO against that pair's combined value $T_{j_1,j_2,a} = V^1_{j_1,a} V^2_{j_2,a}$,

$$dA_{i,j_1,j_2} = \sum_{a=1}^{d_{head}} dO_{i,a}\, T_{j_1,j_2,a}. \tag{F.3}$$

Because $T_{j_1,j_2,a}$ factors into two value rows, no pair term is stored either: the kernel forms the product of those rows on-chip.

F.4 Gradients of the Value Projections

Differentiating the output of Equation (2.7) with respect to the first value set gives

$$dV^1_{j_1,a} = \sum_{i,\, j_2} A_{i,j_1,j_2}\, dO_{i,a}\, V^2_{j_2,a}, \tag{F.4}$$

and, by the same argument applied to the second value set,

$$dV^2_{j_2,a} = \sum_{i,\, j_1} A_{i,j_1,j_2}\, dO_{i,a}\, V^1_{j_1,a}. \tag{F.5}$$

Each value gradient carries the other value set as a factor. That coupling has no counterpart in standard attention and is what the normalization of Section 3.1 controls.

F.5 Gradient of the Scores

The softmax of Equation (2.7) is taken jointly over the pair (j_1, j_2), so its Jacobian couples every pair of a query to every other,

$$\frac{\partial A_{i,j_1,j_2}}{\partial S_{i,j_1',j_2'}} = A_{i,j_1,j_2}\Big(\mathbf{1}\big[(j_1, j_2) = (j_1', j_2')\big] - A_{i,j_1',j_2'}\Big). \tag{F.6}$$

Here $\mathbf{1}[\,\cdot\,]$ is the indicator, equal to one when its condition holds and zero otherwise. It occupies the place of the Kronecker delta in the Jacobian of an ordinary softmax over a single index, and is written as a condition rather than a subscript because the index here is the pair (j_1, j_2). Contracting Equation (F.6) with Equation (F.3) over (j_1, j_2) leaves one subtraction per query,

$$dS_{i,j_1,j_2} = A_{i,j_1,j_2}\big(dA_{i,j_1,j_2} - \delta_i\big), \tag{F.7}$$

where δ_i is the attention-weighted mean of the weight gradients,

$$\delta_i = \sum_{j_1',\, j_2'} A_{i,j_1',j_2'}\, dA_{i,j_1',j_2'}. \tag{F.8}$$

Written this way, δ_i would require the attention tensor $\mathbf{A}$, the third-order tensor the kernel is avoiding. Substituting Equation (F.3) and Equation (2.7) collapses it to a dot product of two rows already in memory,

$$\delta_i = \sum_{a=1}^{d_{head}} dO_{i,a}\, O_{i,a}. \tag{F.9}$$

The kernel computes Equation (F.9) once per query before the backward launches and reuses it in all of them. The ALiBi3D bias enters the score additively and has no learnable parameters, so it contributes nothing further to dS; it is simply re-added whenever a score is recomputed.

F.6 Gradients of the Query and Key Projections

The score of Equation (2.6) is trilinear in Q, K^1, and K^2, so each of these gradients contracts dS against the other two factors. For the query,

$$dQ_{i,a} = \frac{1}{\sqrt{d_{head}}} \sum_{j_1,\, j_2} dS_{i,j_1,j_2}\, K^1_{j_1,a}\, K^2_{j_2,a}. \tag{F.10}$$

For the first key set,

$$dK^1_{j_1,a} = \frac{1}{\sqrt{d_{head}}} \sum_{i,\, j_2} dS_{i,j_1,j_2}\, Q_{i,a}\, K^2_{j_2,a}, \tag{F.11}$$

and for the second,

$$dK^2_{j_2,a} = \frac{1}{\sqrt{d_{head}}} \sum_{i,\, j_1} dS_{i,j_1,j_2}\, Q_{i,a}\, K^1_{j_1,a}. \tag{F.12}$$

The normalizer $\sqrt{d_{head}}$ of Equation (2.6) appears in all three, and in neither value gradient, because it scales the score rather than the output.

F.7 Why Three Kernels Suffice

Two properties of these expressions fix the kernel design of Section 4.

The first is symmetry. Exchanging (K^1, V^1) with (K^2, V^2) and the windows w_1 with w_2 maps Equation (F.4) to Equation (F.5) and Equation (F.11) to Equation (F.12). The exchange transposes the attention tensor as well, since A_{i,j_1,j_2} is built from $K^1_{j_1}$ and $K^2_{j_2}$ and is not symmetric in its two key indices. One kernel organized around blocks of K^1 positions therefore serves for both key–value pairs: run as written it produces dK^1 and dV^1, and run with the pairs exchanged it produces dK^2 and dV^2.

The second is ownership. Equations (F.4) and (F.11) sum over the query index i and over the opposite key index, so a program that owns a block of K^1 positions accumulates complete rows of dK^1 and dV^1 and writes them once. Equation (F.10) instead sums over both key indices at fixed i, so complete rows of dQ are owned by a program that holds a block of queries, which is the dedicated query kernel. Folding dQ into a key-organized kernel would instead place its accumulation inside the scalar K^2 loop, so each query block takes one atomic update per K^2 step, roughly w_2 of them per K^1 block. Every K^1 block then contends on the same dQ rows, so contention grows with w_2, the very window the coverage fix frees. The query kernel accumulates dQ privately in fp32 and writes it once with a plain store, removing those atomic updates at the cost of one extra recomputation of the scores. Each of the three launches recomputes the scores it needs from Equations (2.6) and (F.2), which is the recomputation the split pays for.

G Additional Results: Fitted Scaling Laws

G.1 Coefficient Estimation

We estimate the coefficients of Equation (5.1) following the procedure of Busbridge et al. (2025), adapted to our operators. The supervised scaling law is, restated for convenience,

$$L(N, D) = E + \left(\frac{A}{N^\alpha} + \frac{B}{D^\beta}\right)^\gamma, \tag{G.1}$$

where E is the irreducible loss. To aid numerical stability, we fit in log space. For $a, b > 0$,

$$\log(a + b) = \log\left(\exp\log a + \exp\log b\right) = \mathrm{LSE}(\log a, \log b), \tag{G.2}$$

where LSE is the log-sum-exp operator. The law in log form is

$$\log L(N, D; A, B, E, \alpha, \beta, \gamma) = \log\left[E + \left(\frac{A}{N^\alpha} + \frac{B}{D^\beta}\right)^\gamma\right] \tag{G.3}$$

$$= \mathrm{LSE}\left[\log E,\ \gamma \log\left(\frac{A}{N^\alpha} + \frac{B}{D^\beta}\right)\right] \tag{G.4}$$

$$= \mathrm{LSE}\left[\log E,\ \gamma\,\mathrm{LSE}\left(\log A - \alpha \log N,\ \log B - \beta \log D\right)\right]. \tag{G.5}$$

We make no assumptions about the relationships between the coefficients, and we fit each operator independently. We minimize

$$(A^*, B^*, E^*, \alpha^*, \beta^*, \gamma^*) = \underset{\{A,B,E,\alpha,\beta,\gamma\}}{\arg\min} \sum_i \mathrm{Huber}_\delta\left(\log L(N^{(i)}, D^{(i)}; A, B, E, \alpha, \beta, \gamma) - \log L^{(i)}\right) \tag{G.6}$$

with a Huber $\delta = 10^{-4}$, where $N^{(i)}$, $D^{(i)}$, and $L^{(i)}$ are the model size, number of training tokens, and loss of the i-th run. We fit on the 21 IsoFLOP points of each operator over a grid of L-BFGS-B initializations given by $\log A \in \{0, 5, 10, 15, 20\}$, $\log B \in \{0, 5, 10, 15, 20\}$, $\log E \in \{-1, -0.5, 0, 0.5, 1, 1.5\}$, $\alpha \in \{0, 0.5, 1, 1.5\}$, $\beta \in \{0, 0.5, 1, 1.5\}$, $\gamma \in \{0, 0.5, 1, 1.5\}$. We obtain confidence intervals by a seeded nonparametric bootstrap of 2048 resamples.

G.2 Fitted Coefficients

Fitting Equation (5.1) to the 21 IsoFLOP points of each architecture at $n_{ctx} = 512$ gives

$$L_{\text{standard}}(N, D) = 2.088 + \left(\frac{5.883 \times 10^5}{N^{0.719}} + \frac{5.741 \times 10^7}{D^{0.833}}\right)^{0.618}, \tag{G.7}$$

$$L_{\text{higher-order}}(N, D) = 1.955 + \left(\frac{1.971 \times 10^5}{N^{0.656}} + \frac{8.907 \times 10^5}{D^{0.640}}\right)^{0.614}, \tag{G.8}$$

with coefficients and bootstrap confidence intervals in Table G.1; both fits have mean relative error below 0.2%.

The floor difference is $\Delta E = 0.134$ nats. Under a paired bootstrap its 95% interval is $[0.007, 0.358]$ with $P(\Delta E > 0) = 0.979$; bootstrapping the two fits independently gives $[-0.040, 0.403]$ with $P(\Delta E > 0) = 0.937$.

The compute-optimal allocation $N_{\text{opt}} \propto C^a$, with $a = \beta/(\alpha + \beta)$, is $a = 0.537$ $[0.508, 0.562]$ for standard attention and $a = 0.494$ $[0.445, 0.525]$ for higher-order attention.

Table G.1: Fitted coefficients of Equation (5.1) at $n_{ctx} = 512$, with 95% bootstrap confidence intervals (2048 resamples); 21 IsoFLOP points per architecture. Mean relative error of the fits is 0.18% (standard) and 0.19% (higher-order).

	Standard	Higher-order
E	2.088 $[1.995, 2.176]$	1.955 $[1.753, 2.075]$
α	0.719 $[0.680, 0.763]$	0.656 $[0.603, 0.723]$
β	0.833 $[0.796, 0.852]$	0.640 $[0.594, 0.654]$
γ	0.618 $[0.567, 0.694]$	0.614 $[0.529, 0.693]$
A	5.883×10^5	1.971×10^5
B	5.741×10^7	8.907×10^5

G.3 Comparison with the Parameter-Exponent Fit of Roy et al.

Roy et al. (2025) also report that higher-order attention changes a scaling exponent, but fit a different quantity from ours, so the two exponents are not directly comparable. They train three Mixture-of-Experts (MoE) models, from 1B active (57B total) to 3.5B active (176B total) parameters, with a windowed higher-order operator in every fourth layer, at a fixed token budget. They fit downstream loss directly as a log-linear function of active parameters,

$$-\log L(N) = \alpha \log N + b, \tag{G.9}$$

over the three models, separately for each downstream benchmark, using downstream negative log-likelihood for L and active parameters for N, with b the fitted intercept. Under this fit the parameter exponent α is steeper for higher-order attention on every benchmark (Table G.2).[3]

Table G.2: Parameter-exponent estimates of Roy et al. (2025) under Equation (G.9), fit over three MoE models at a fixed token budget on downstream negative log-likelihood. Reproduced from their Table 3; Δ is the relative increase for higher-order attention.

α	GSM8K	MMLU	MMLU-pro	MBPP
Standard	0.1420	0.1256	0.0901	0.1720
Higher-order	0.1683	0.1364	0.1083	0.1837
Δ	+18.5%	+8.5%	+20.2%	+6.8%

Three reasons keep the two results from conflicting, summarized in Table G.3.

First, the fitted quantity differs: Equation (G.9) estimates the *parameter* exponent α, whereas our floor comparison rests on the irreducible loss E and the *data* exponent β, which we find is smaller

[3] The MMLU increase reproduces the +8.5% reported in their Table 3; the ratio of the two exponents shown rounds to +8.6%.

Table G.3: How the fit of Roy et al. (2025) differs from ours.

	Roy et al. (2025)	This work
Architecture	MoE, windowed, every fourth layer	dense, every layer
Models	three sizes, 1–3.5B active	IsoFLOP sweep, 21 points
Law	$-\log L(N) = \alpha \log N + b$	$L(N, D) = E + \left(A/N^{\alpha} + B/D^{\beta}\right)^{\gamma}$
Fitted quantity	parameter exponent, per benchmark	data exponent β and floor E
Metric	downstream negative log-likelihood	held-out cross-entropy

for higher-order attention, not larger. Second, the setup differs: their fit uses MoE models with the operator in every fourth layer, evaluated on downstream benchmarks, against our dense models with the operator in every layer, evaluated on held-out cross-entropy. Third, the fit rests on three points per benchmark; a two-parameter line through three points nearly always achieves a high R^2 (> 0.996 in their Table 4), so the fit quality says little about the confidence interval on each slope. The two studies agree on the direction, that higher-order attention improves scaling, while measuring it along different axes.

H Additional Results: Downstream Evaluation

Section 7 reports the composite accuracy–cross-entropy curve; here we give the benchmarks, the model set, and the per-task curves behind it (Figure H.1).

We evaluate six benchmarks: ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MathQA (Amini et al., 2019), PIQA (Bisk et al., 2019), and WinoGrande (Sakaguchi et al., 2019). These benchmarks require no generation, so they compare architectures without depending on the decoding path. The models are those of Section 5.1 and Section 6.2: standard attention alongside the dense, windowed, and dilated higher-order variants. Each point is one model, placed at its measured cross-entropy and its mean accuracy over the six tasks. Cross-entropy is not comparable across context lengths, so most points come from the $n_{ctx} = 512$ models.

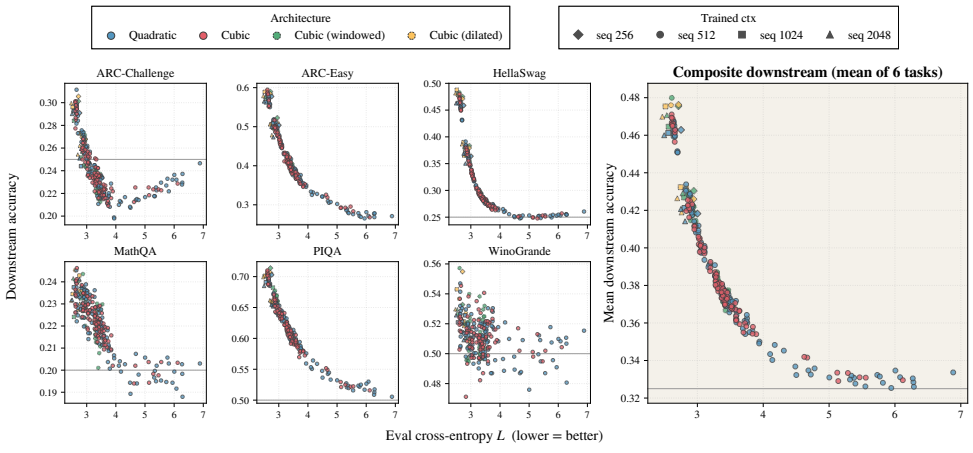
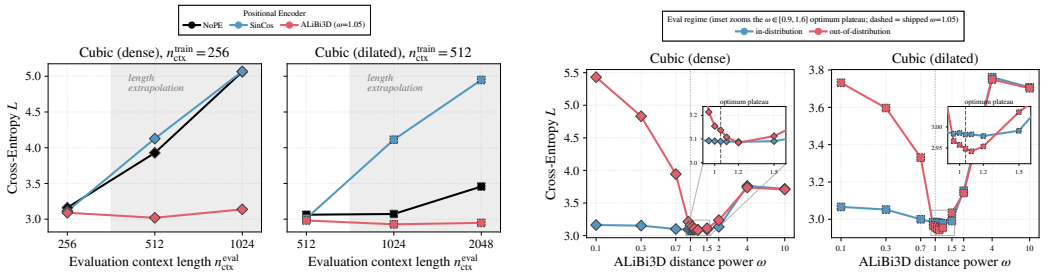


Figure H.1: **Per-task downstream accuracy against pretraining cross-entropy.** Standard attention and dense, windowed, and dilated higher-order attention on the six composite benchmarks (ARC-Easy, ARC-Challenge, HellaSwag, MathQA, PIQA, WinoGrande) and their mean; each point is one model. On every task, and in the mean, all architectures fall on one shared accuracy–cross-entropy curve.

I Additional Results: Positional Encoding under Dilated Attention

With no positional encoding, the dense operator collapses under extrapolation while the dilated one partially holds (Figure I.1a): the strided gather ties the tokens a query sees to its position, a relative-



(a) Length extrapolation, dense and dilated.

(b) Distance exponent ω, dense and dilated.

Figure I.1: **Positional encoding under dilated attention.** The dense and dilated counterparts of the length-extrapolation and distance-exponent studies of Section 3.2 (Figures 3.1b and 3.1c).

position structure that removing the encoder leaves intact (Kazemnejad et al., 2023), though the dense K^2 leg still grows with evaluation length and eventually gives way. The distance exponent ω behaves as it does for the dense operator (Figure I.1b): in distribution loss is largely flat, while out of distribution loss is lowest just above $\omega = 1$.

J Dilated Higher-Order Attention: Construction and Cost

This section gives the dilated construction of Section 6.1 in full: how a tier is built, how the tiers are merged, and what the schedule costs relative to dense higher-order attention. We adapt LongNet's multiscale dilation (Ding et al., 2023), which cuts the sequence into segments and keeps a strided subset of each, so that a single pass sees nearby positions finely and distant ones coarsely. Each tier is defined by a segment width w and stride r: within each segment, only every r-th position is retained. Small segments provide fine resolution over nearby positions, while larger segments provide coarser coverage of increasingly distant positions.

Asymmetric dilation. For higher-order attention, we apply this dilation asymmetrically. At each tier, we gather strided subsets of Q, K^1, and V^1, while keeping K^2 and V^2 dense and causal. This reduces the resolution of the Q-K^1 interaction while preserving global access through K^2 and V^2: every query still reaches every position, but its interaction with K^1 is evaluated at different resolutions across tiers.

Aggregating across tiers. Each tier performs an independent attention computation over its gathered subset. For each query position i, it produces an output $O_{t,i}$ together with the accumulated softmax denominator $Z_{t,i}$. We combine the tiers by normalizing these denominators across tiers:

$$O_i = \sum_t \pi_{t,i} O_{t,i}, \qquad \pi_{t,i} = \frac{Z_{t,i}}{\sum_{t'} Z_{t',i}}. \tag{J.1}$$

Each tier contributes according to the fraction of the total attention mass it accounts for at that query position. This is equivalent to combining partial softmax computations across blocks, as in FlashAttention. Since the tiers are independent until this final merge, they can be computed in parallel without introducing a sequential dependency.

Schedule and cost. We choose the tier schedule so that every tier gathers the same number of positions. Let $G = w/r$ denote the number of positions retained from each segment. We double the segment width at each successive tier and increase the stride proportionally:

$$(w_t, r_t) = (2^t G,\ 2^t). \tag{J.2}$$

The gathered length therefore remains fixed at G, while the receptive field of each tier doubles. We continue until $w_t = n_{ctx}$, giving $\log_2(n_{ctx}/G) + 1$ tiers.

Within every tier each gathered query does the same work, so a tier's cost scales with the number of queries it gathers. Tier t gathers n_{ctx}/r_t queries, each attending to G candidates for K^1 and the full

context for K^2, so before causal masking the dilated cost relative to dense higher-order attention is

$$\frac{G}{n_{ctx}} \sum_t \frac{1}{r_t} = \frac{2G}{n_{ctx}} - \frac{G^2}{n_{ctx}^2} < \frac{2G}{n_{ctx}}, \tag{J.3}$$

which is the unmasked dilated pair fan-in $N_{\text{blk}}\, G^2$ of Section K divided by the unmasked dense fan-in n_{ctx}^2.

Causal masking does not cancel between the two. Dense higher-order attention bounds both key axes by the same query position, so its causal average is $n_{ctx}^2/3$ rather than $n_{ctx}^2/4$, whereas the dilated variant bounds its two axes independently, one within a segment of G gathered positions and one over the full context, and each halves. Carrying both factors through Equation (J.3) gives

$$\frac{C_{\text{dilated}}}{C_{\text{dense}}} = \frac{\bar{f}_{\text{dil}}}{\bar{f}_{\text{cub}}} \approx \frac{3G}{2n_{ctx}}, \tag{J.4}$$

a speedup of roughly $2n_{ctx}/(3G)$, which is the accounting used for the FLOPs we report (Equations (K.9) and (K.14)). For example, with $G = 64$ and $n_{ctx} = 2048$, the schedule uses six tiers and reduces the attention cost by about 21×.

K FLOP and Parameter Comparison: Standard vs. Higher-Order Attention

K.1 Theoretical FLOP Comparison

Theoretical forward FLOPs per token, comparing standard attention against full-window higher-order attention, following the per-token, read-off-notation convention of Busbridge et al. (2025): no GQA, and an MLP of n_{ffn} weight matrices each of width $d_{ffn} = \rho_{ffn} d_{model}$, taking the standard two-matrix case $n_{ffn} = 2$ throughout this subsection. Counts are per token: multiply by batch $\times n_{ctx}$ for a batch, and by n_{layer} for the stack. Only the attention sublayer changes between the two: higher-order attention replaces the query–key dot product with a three-way product against *two* key sets, and the value read-out with a Hadamard product of *two* value sets (Section 3).

Why the causal fan-in differs. In standard attention, query i attends to i keys, averaging to $n_{ctx}/2$ causal keys per token. In higher-order attention, query i attends to the i^2 key *pairs* (j, k) with $j, k \le i$, averaging to $\approx n_{ctx}^2/3$ causal pairs per token – one third of the dense n_{ctx}^2, generalizing the standard half. This extra factor of n_{ctx} in the fan-in, together with a three-factor (rather than two-factor) logit/value product, is what drives the context-dependent terms below from linear to quadratic in n_{ctx}.

Table K.1: Per-token forward FLOPs of a pre-LN Transformer block with full-window higher-order attention, by operation.

Operation	FLOPs / token	Order
Input projections (Q, K^1, K^2, V^1, V^2)	$10 d_{model}^2$	$O(d_{model}^2)$
Logits $\sum_e QK^1K^2$	$d_{model} n_{ctx}^2$	$O(d_{model} n_{ctx}^2)$
Softmax over $j, k \le i$	$\approx \frac{4}{3} n_{head} n_{ctx}^2$	$O(n_{head} n_{ctx}^2)$
Value contraction $\sum_{j,k \le i} SV^1V^2$	$d_{model} n_{ctx}^2$	$O(d_{model} n_{ctx}^2)$
Output projection W^O	$2 d_{model}^2$	$O(d_{model}^2)$
MLP (W^1, ϕ, W^2)	$4 d_{model} d_{ffn}$	$O(d_{model} d_{ffn})$
Total / token	$12 d_{model}^2 + 4 d_{model} d_{ffn} + 2 d_{model} n_{ctx}^2 + \frac{4}{3} n_{head} n_{ctx}^2$	

With $d_{ffn} = \rho_{ffn} d_{model}$, the matmul cost per token is

$$\text{higher-order:} \quad 12 d_{model}^2 + 4 d_{model} d_{ffn} = \left(12 + 4\rho_{ffn}\right) d_{model}^2 = 28 d_{model}^2, \tag{K.1}$$

$$\text{standard:} \quad 8 d_{model}^2 + 4 d_{model} d_{ffn} = \left(8 + 4\rho_{ffn}\right) d_{model}^2 = 24 d_{model}^2, \tag{K.2}$$

where the rightmost column evaluates at $\rho_{ffn} = 4$. Summing over n_{layer} blocks and adding the shared embedding/final-logits terms:

$$C_{\text{fwd}} \approx 2N + 2 n_{layer} d_{model} n_{ctx}^2 + \tfrac{4}{3} n_{layer} n_{head} n_{ctx}^2 + 2 n_{vocab} d_{model}, \tag{K.3}$$

where the non-embedding parameter count is

$$N = n_{layer} d_{model}^2 \left(6 + n_{ffn} \rho_{ffn}\right), \tag{K.4}$$

The attention-parameter coefficient rises from the standard 4 (three input projections + W^O) to 6 (five input projections + W^O), and the context term becomes quadratic rather than linear in n_{ctx}.

Table K.2: Per-token forward FLOPs for one transformer layer under different accounting conventions and attention orders. Differences from Kaplan et al. (2020)'s standard convention are shown alongside the causal-accounted standard and higher-order columns.

FLOPs	Kaplan et al. (2020) (standard)	Standard (causal)	Higher-order (causal, ours)
Input proj.	$6d_{model}^2$	$6d_{model}^2$	$10d_{model}^2$
Logits	$2d_{model}n_{ctx}$	$d_{model}n_{ctx}$	$d_{model}n_{ctx}^2$
Softmax	(not counted)	$2n_{head}n_{ctx}$	$\frac{4}{3}n_{head}n_{ctx}^2$
Values	(in Logits)	$d_{model}n_{ctx}$	$d_{model}n_{ctx}^2$
Project	$2d_{model}^2$	$2d_{model}^2$	$2d_{model}^2$
MLP	$4\rho_{ffn}d_{model}^2$	$4\rho_{ffn}d_{model}^2$	$4\rho_{ffn}d_{model}^2$
Attention total	$8d_{model}^2 + 2d_{model}n_{ctx}$	$8d_{model}^2 + 2d_{model}n_{ctx} + 2n_{head}n_{ctx}$	$12d_{model}^2 + 2d_{model}n_{ctx}^2 + \frac{4}{3}n_{head}n_{ctx}^2$

Under matched causal conventions (Table K.2), swapping standard for full-window higher-order attention multiplies the context-dependent term by n_{ctx} (from $2d_{model}n_{ctx}$ to $2d_{model}n_{ctx}^2$) and the attention parameters by $\frac{3}{2}$ (from $4d_{model}^2$ to $6d_{model}^2$), while the MLP, (de-)embedding, and all subleading terms are untouched. Local windows $w_1 \times w_2 < n_{ctx}^2$ (as used in practice, Section 6.1) recover a context term that is constant in n_{ctx},

$$6n_{layer}d_{model}\, w_1 w_2, \tag{K.5}$$

and comparing this against the $2N$ of Equation (K.3) restores $6N$-style accounting whenever

$$w_1 w_2 \ll \frac{\left(6 + n_{ffn}\rho_{ffn}\right) d_{model}}{3}. \tag{K.6}$$

K.2 Parameter Accounting

The idealized N of Equation (K.4) is convenient for comparing architectures, but the models we train depart from it in ways that matter once N is the independent variable of a scaling law. We therefore state the count exactly.

Both architectures share a vocabulary of $n_{vocab} = 100{,}277$, a gated Swish-Gated Linear Unit (SwiGLU) MLP with $d_{ffn} = \lfloor 2.75\, d_{model} \rfloor$ and three weight matrices, and two RMSNorm(d_{model}) weights per block. Both use *untied* embeddings: a lookup on the input side, and a separate biased Linear(d_{model}, n_{vocab}) with its own prenorm on the output side. The theoretical comparison of Section K.1 instead assumes tied embeddings, $n_{vocab} = 32{,}768$, and a two-matrix MLP ($n_{ffn} = 2$); the trained models differ in all three. The two architectures differ only in the attention sublayer.

With P from Table K.3, the totals are

$$N_{\text{total}} = \underbrace{n_{vocab}d_{model}}_{\text{input embedding}} + n_{layer}P + \underbrace{n_{vocab}d_{model} + n_{vocab} + d_{model}}_{\text{output projection, bias, prenorm}}, \qquad N = N_{\text{total}} - n_{vocab}d_{model}, \tag{K.7}$$

so N excludes only the input embedding. Because the embeddings are untied, the output projection is a genuine weight matrix rather than a transpose of the input table, and it is counted in N.

Equation (K.7) agrees with the realized parameter counts of every model in our sweeps, spanning $n_{layer} = n_{head} \in \{8, \ldots, 22\}$ with $d_{model} = 128\, n_{layer}$, to within 0.001%.

Table K.3: Per-block learnable parameter count P. Standard attention uses a fused query–key–value projection and an output projection; higher-order attention uses five input projections and an output projection. Higher-order key-norm gains are frozen and excluded here, so a count that includes them is $2d_{head}$ per block larger.

Component	Standard	Higher-order
Attention projections	$4d_{model}^2$	$6d_{model}^2$
QK-norm gains	$2d_{head}$	d_{head}
V-norm gains	—	$2d_{head}$
SwiGLU MLP	$3d_{model}d_{ffn}$	
Block Root-Mean-Square Normalizations (RMSNorms)	$2d_{model}$	
Total P	$4d_{model}^2 + 3d_{model}d_{ffn} + 2d_{model} + 2d_{head}$	$6d_{model}^2 + 3d_{model}d_{ffn} + 2d_{model} + 3d_{head}$

K.3 Sparse Variants

The windowed and dilated variants of Section 6 differ from dense higher-order attention in exactly one quantity: the average number of key *pairs* a query attends to. Writing that per-token fan-in as $\bar{f}$, the projections, output projection and MLP are untouched, and the whole per-block cost reduces to

$$F_{\text{block}} = 12d_{model}^2 + n_{ffn} \cdot 2d_{model}d_{ffn} + \underbrace{6d_{model}\,\bar{f}}_{\text{logits and values}} + \underbrace{4n_{head}\,\bar{f}}_{\text{softmax}}, \tag{K.8}$$

so the three variants are three values of a single scalar. Dense higher-order attention averages i^2 causal pairs over the sequence, which sums exactly:

$$\bar{f}_{\text{cub}} = \frac{1}{n_{ctx}} \sum_{i=1}^{n_{ctx}} i^2 = \frac{(n_{ctx}+1)(2n_{ctx}+1)}{6} \approx \frac{n_{ctx}^2}{3}. \tag{K.9}$$

Its leading order recovers the context-dependent rows of Table K.1.

Windowed. Query i sees two independent causal windows, of widths w_1 and w_2, so of the i^2 pairs available to it, it attends to $\min(i, w_1) \cdot \min(i, w_2)$. Writing $a = \min(w_1, w_2)$ and $b = \max(w_1, w_2)$ and averaging over positions in the three regimes where neither, one, or both windows have saturated gives

$$\bar{f}_{\text{win}} = \frac{1}{n_{ctx}} \left[\frac{a(a+1)(2a+1)}{6} + a\left(\frac{b(b+1)}{2} - \frac{a(a+1)}{2} \right) + ab\,(n_{ctx} - b) \right]. \tag{K.10}$$

Its order in n_{ctx} depends only on how many of the two widths grow with the sequence. With both bounded the last term dominates,

$$\bar{f}_{\text{win}} \approx ab, \tag{K.11}$$

constant in n_{ctx} and so cheaper in order than standard attention. With one left dense, $b = n_{ctx}$, the last term vanishes and the second dominates,

$$\bar{f}_{\text{win}} \approx \frac{an_{ctx}}{2}, \tag{K.12}$$

linear in n_{ctx} and therefore the same order as standard attention, scaled by the bounded width. With both dense, $a = b = n_{ctx}$, Equation (K.10) reduces to $\bar{f}_{\text{cub}}$ by substitution rather than approximation. Windowing thus interpolates between the two baselines according to how many windows are left unbounded, and since it can only remove pairs the dense case counts, $\bar{f}_{\text{win}} \le \bar{f}_{\text{cub}}$ always.

Dilated. Gather and scatter are indexing rather than arithmetic, so only the attention actually computed counts. Every tier decomposes into self-contained mini attentions of identical shape $G \times G \times n_{ctx}$: G queries, G candidate K^1 positions within the segment, and the full dense n_{ctx} for K^2. A tier of width $w_t = 2^t G$ contributes n_{ctx}/w_t segments, and the schedule of Section J runs $T = \log_2(n_{ctx}/G) + 1$ tiers, so the geometric series gives

$$N_{\text{blk}} = \sum_{t=0}^{T-1} \frac{n_{ctx}}{w_t} = \frac{2n_{ctx}}{G} - 1 \tag{K.13}$$

such blocks in total. Causal masking then halves each of the two key axes independently, so

$$\bar{f}_{\text{dil}} = \frac{N_{\text{blk}}\, G^2}{4}. \tag{K.14}$$

This treats every segment as masking alike regardless of position, an approximation that tightens as n_{ctx}/G grows. Dropping the -1 of Equation (K.13) leaves

$$\bar{f}_{\text{dil}} \approx \frac{G n_{ctx}}{2}, \tag{K.15}$$

which is the windowed mixed regime of Equation (K.12) with $a = G$. The two schemes coincide at leading order because both are a fixed-width causal axis contracted against a dense causal one; dilation reaches that fixed width through N_{blk} tiled blocks rather than one sliding window, which is what lets it retain a global view while windowing does not (Section 6.1).

K.4 Summary

Table K.4 collects the four accountings. Standard attention is the only one whose fan-in is a single index rather than a pair, which is why its context term is linear where the others are governed by $\bar{f}$ through Equation (K.8). Among the higher-order variants only the value of $\bar{f}$ changes: dense attends to every causal pair, windowing bounds both key axes, and dilation bounds the K^1 axis at G while leaving K^2 dense.

Table K.4: Per-block forward FLOPs per token for standard attention and the three higher-order variants, at leading order in n_{ctx}. All four share the projection and MLP terms; they differ only in the context-dependent terms, which for the higher-order variants are set by the per-token pair fan-in $\bar{f}$ of Equation (K.8). Windowed uses $a = \min(w_1, w_2)$, $b = \max(w_1, w_2)$; dilated uses gathered length G.

Variant	Proj.	Pair fan-in $\bar{f}$	Context terms	Order in n_{ctx}
Standard	$8d^2_{model}$	— (single key)	$2d_{model}n_{ctx}+2n_{head}n_{ctx}$	$O(n_{ctx})$
Higher-order, dense	$12d^2_{model}$	$\approx n^2_{ctx}/3$	$2d_{model}n^2_{ctx} + \frac{4}{3}n_{head}n^2_{ctx}$	$O(n^2_{ctx})$
Higher-order, windowed	$12d^2_{model}$	Equation (K.10); $\approx ab$ or $an_{ctx}/2$	$6d_{model}\bar{f}_{\text{win}} + 4n_{head}\bar{f}_{\text{win}}$	$O(1)$ to $O(n^2_{ctx})$
Higher-order, dilated	$12d^2_{model}$	$N_{\text{blk}}G^2/4$, $N_{\text{blk}} = 2n_{ctx}/G - 1$; $\approx Gn_{ctx}/2$	$6d_{model}\bar{f}_{\text{dil}} + 4n_{head}\bar{f}_{\text{dil}}$	$O(n_{ctx})$

Three relations are worth reading off the table. Both sparse variants satisfy $\bar{f} \le \bar{f}_{\text{cub}}$ for any setting of their parameters, since each attends to a subset of the causal pairs the dense operator counts and neither adds any, so neither performs more attention-interior arithmetic than the dense operator. Windowing spans the widest range: with both windows bounded it is cheaper in order than standard attention, and with both left dense it reduces exactly to the dense higher-order row. And dilation sits at the same order as standard attention, $O(n_{ctx})$. Its pair fan-in exceeds standard attention's key fan-in by a factor G, comparing $\bar{f}_{\text{dil}} \approx Gn_{ctx}/2$ against $(n_{ctx}+1)/2$, and because each pair costs a three-factor rather than a two-factor product, the dominant d_{model} term of the context cost grows by roughly $3G/2$. Remaining linear in n_{ctx} is what makes the variant affordable at the sequence lengths of Section 6.2.

Figure K.1 evaluates these accountings numerically, sweeping each variant's compute–coverage knob at the shipped 8-block backbone ($\approx$ 0.21B non-embedding parameters) and reporting the per-token forward-FLOPs saving over dense higher-order attention. The saving grows with context because the dense cost it is measured against is $O(n^2_{ctx})$ while both sparse variants are bounded below it. At the shipped dilated setting $G = 64$, the dilated variant costs 4.0× less per token than dense at $n_{ctx} = 512$, 19× less at $n_{ctx} = 2048$, and 83× less at $n_{ctx} = 8192$; the shipped windowed setting $w_1/w_2 = 256/64$ saves 4.6×, 58×, and 891× at the same three contexts. Crucially, the dilated saving is bounded relative to standard attention rather than unbounded: at $n_{ctx} = 2048$ the dilated variant costs 8.2× standard attention (rising to 24× at $n_{ctx} = 8192$), a fixed multiple set by the three-factor product and the gathered length G, whereas its saving over the $O(n^2_{ctx})$ dense operator diverges. Dilation therefore buys higher-order expressiveness at standard attention's asymptotic order, which is what keeps it affordable at long context.

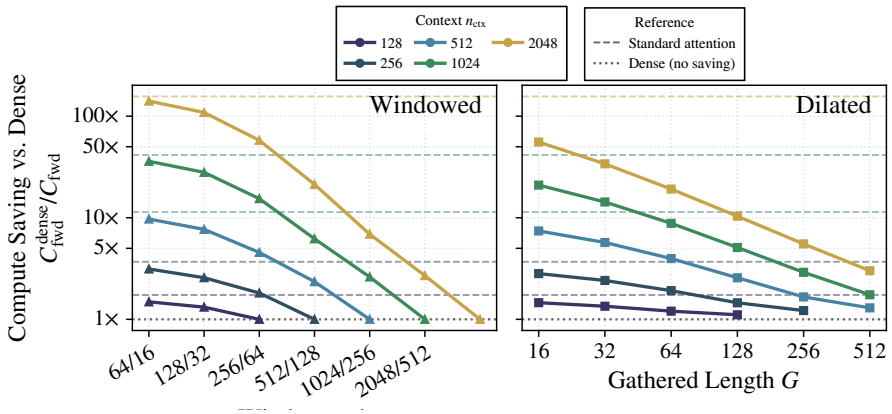


Figure K.1: **Per-token compute saved by the sparse variants, relative to dense higher-order attention.** Forward-FLOPs saving $C_{\text{fwd}}^{\text{dense}}/C_{\text{fwd}}$ (Section K) as the compute–coverage knob is swept, for the windowed variant (left, by window widths w_1/w_2) and the dilated variant (right, by gathered length G), at 8 blocks and each trained context length n_{ctx}. Dashed lines mark the standard-attention ceiling ($C_{\text{fwd}}^{\text{dense}}/C_{\text{fwd}}^{\text{std}}$, the most any variant could save); the dotted line is dense attention (1×). The saving grows with context: at $n_{ctx} = 2048$ the dilated variant at $G = 64$ is 19× cheaper per token than dense (the windowed 256/64 setting 58×), and both exceed two orders of magnitude beyond $n_{ctx} = 4096$. Values are analytic, not measured.

L Experimental Setting

This section collects the settings behind the experiments, beyond the summary of Section 2.4. The shared per-token FLOPs and parameter conventions are given in Section K; the model family is listed in Table L.1, and the fitting procedure in Section G.

Model family. All models are pre-norm, decoder-only dense Transformers (Grattafiori et al., 2024). Every model has a fixed aspect ratio $d_{model} = 128\, n_{layer}$ and a fixed head dimension $d_{head} = 128$, so the head count is $n_{head} = n_{layer}$ and the per-head width is the same at every size. The FFNs are gated SwiGLU (Shazeer, 2020) with ratio $\rho_{ffn} = 2.75$ ($n_{ffn} = 3$ weight matrices, $d_{ffn} = \lfloor 2.75\, d_{model} \rfloor$), and normalization is RMSNorm ($\epsilon = 10^{-5}$) with a learnable query–key gain (`qk_norm`). Input and output embeddings are untied, and the vocabulary is $n_{vocab} = 100{,}277$ (the `cl100k_base` tokenizer). Standard (quadratic) attention uses $4d_{model}^2$ projection parameters per layer; higher-order (cubic) attention uses $6d_{model}^2$, from the extra K^2, V^2 projections (Section K). The IsoFLOP sweep of Section 5.1 spans $n_{layer} \in \{11, 13, 15, 17, 19\}$ (d_{model} from 1408 to 2432, up to $N \approx 1.6$B non-embedding parameters for standard and ≈ 1.8B for higher-order attention) and trains at $n_{ctx} = 512$; the sparse-context and frontier experiments (Sections 6.2 and 6.3) also use $n_{layer} = 8$ and $n_{layer} = 13$ and train at $n_{ctx} = 2048$. Table L.1 lists each size. The IsoFLOP sweep uses three seeds per cell, and the scaling laws are fit to the mean end-of-training cross-entropy over those seeds; the sparse-context and frontier experiments use a single seed per cell.

Data. We train on the high-quality split of Nemotron-CC (Su et al., 2025), streamed and packed to the training context length with no padding. Nemotron-CC is built from 99 English Common Crawl snapshots (CC-MAIN-2013-20 through CC-MAIN-2024-30) with global fuzzy and exact-substring deduplication, giving a 6.3T-token corpus (4.4T deduplicated real tokens and 1.9T synthetic). Documents are scored by an ensemble of three quality classifiers and grouped into five quality levels; the high-quality split (Nemotron-CC-HQ) is the 1.1T-token subset of the highest-scoring real tokens (0.6T) and diverse question–answer synthetic tokens (0.5T). Our training budgets are well within a single epoch of this subset, so runs see essentially unique tokens. Sequences are tokenized with `cl100k_base` ($n_{vocab} = 100{,}277$). Held-out cross-entropy is reported on a fixed evaluation sample drawn from the same split with a fixed seed, so every run is scored on identical held-out data. The context-extrapolation and train-context–by–eval-context studies (Section 3.2) average each cell over roughly 10^9 held-out tokens.

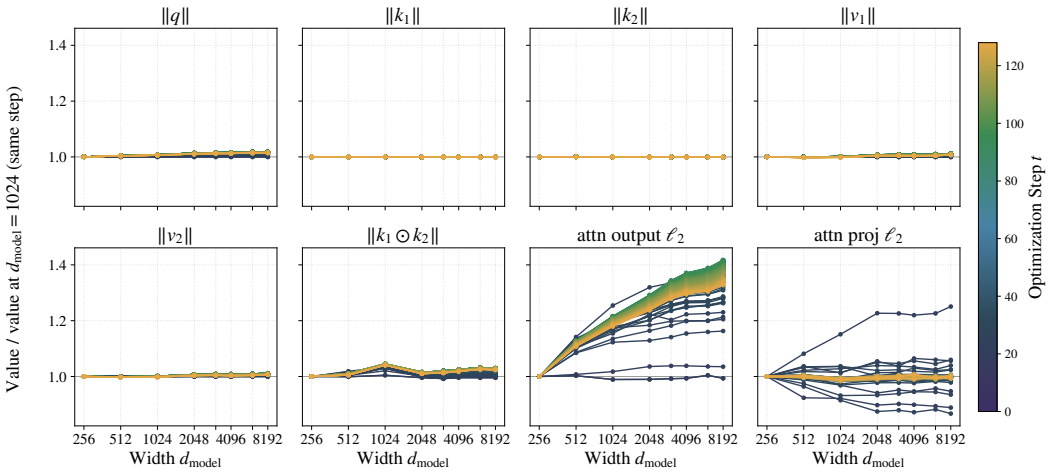


Figure L.1: **Width coordinate check for higher-order attention under CompleteP.** Per-coordinate activation norms of the higher-order attention layer as width d_{model} grows from 256 to 8192, each panel one probed quantity, normalized to its $d_{model} = 1024$ value at the same optimization step (colour, dark to light). Flat traces at 1.0 mean the quantity is width-invariant. Every parametrized coordinate stays flat, confirming that the CompleteP initialization and per-parameter-group Adam with decoupled weight decay (AdamW) learning-rate rules transfer across width. The lone exception, the raw kernel output before the output projection (*attn output*), grows mildly with width because larger models learn a sharper attention distribution that averages over fewer value pairs; the parametrized output projection absorbs this, so the layer output (*attn proj*) is width-invariant.

Table L.1: The model family used in this work. Every size has a fixed aspect ratio $d_{model} = 128\, n_{layer}$, head dimension $d_{head} = 128$ (so $n_{head} = n_{layer}$), gated SwiGLU FFN with $d_{ffn} = \lfloor 2.75\, d_{model} \rfloor$, and vocabulary $n_{vocab} = 100{,}277$. N_{std} and $N_{\text{h.o.}}$ are the *non-embedding* parameter counts (in billions) for standard and higher-order attention, following the convention of Equation (K.7): N excludes only the input embedding, so the untied output projection is counted. The two architectures differ only in the attention projections ($4d^2_{model}$ vs. $6d^2_{model}$ per layer, Section K). The last column marks where each size is used: the IsoFLOP scaling sweep (Section 5.1), the sparse-context sweep (Section 6.2), and the frontier comparison (Section 6.3).

n_{layer}	d_{model}	d_{ffn}	n_{head}	N_{std} (B)	$N_{\text{h.o.}}$ (B)	Used in
8	1024	2816	8	0.206	0.222	context, frontier
11	1408	3872	11	0.408	0.452	IsoFLOP
13	1664	4576	13	0.608	0.680	IsoFLOP, context
15	1920	5280	15	0.870	0.981	IsoFLOP
17	2176	5984	17	1.204	1.365	IsoFLOP
19	2432	6688	19	1.621	1.846	IsoFLOP

Optimization. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$) and decoupled weight decay 0.1 applied to hidden weights only. Gradients are clipped to a global ℓ_2 norm of 1.0, and a z-loss of 10^{-4} is added for stability (Equation (2.2)) (Chowdhery et al., 2022). The learning rate follows a cosine schedule with 5% linear warmup, warm-start value 10^{-6}, and terminal value 10^{-6}. Learning rate and initialization are transferred across model sizes with CompleteP (Dey et al., 2025; Mlodozeniec et al., 2025) at $\alpha = 1.0$, tuned at the base size $d_{model} = 1024$, $n_{layer} = 8$ (base initialization std 0.02). A width coordinate check (Figure L.1) confirms that this transfer holds for higher-order attention: every parametrized coordinate stays width-invariant as d_{model} grows. The base peak learning rate is 0.005 for higher-order attention and 0.003 for standard attention, resolved by per-architecture sweeps at $n_{ctx} = 512$ and re-tuned per context length. Each sweep is a single-seed grid search over $\{0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05, 0.07\}$. All runs hold a constant global batch of 655,360 tokens per step; training length follows a Chinchilla-style $20N$ token budget except in the IsoFLOP sweep, where each run's token count is set by its compute budget as $D = C/6N$, and in the ALiBi3D exponent sweep below, which uses $30N$. Training is in `bfloat16` with FSDP2 and `torch.compile`; higher-order runs use the FlashAttention-2 speedup kernel (Dao, 2023) with an fp32/TF32 score path. All runs are trained on nodes of NVIDIA H100 GPUs.

Hyperparameters of the operator. Higher-order attention uses individual value normalization (Section 3.1): V^1 and V^2 are normalized separately before the paired product. The K^1, K^2 query–key gains are frozen at initialization, leaving the query gain as the sole learnable score temperature; the trilinear logit depends only on the product $\gamma_q \gamma_{k^1} \gamma_{k^2}$, so three independent learnable gains are redundant and drive a γ^3 amplification. Positions are encoded with ALiBi3D, our three-way extension of ALiBi (Press et al., 2022), at exponent $\omega = 1.05$ (Sections 3.2 and E.2). That value comes from a staged sweep at $n_{layer} = 8$ and $n_{ctx} = 512$: the first stage fixes $\omega = 1$ and sweeps the learning rate to 0.005, and the second fixes the learning rate there and grid-searches $\omega \in \{1.0, 1.05, 1.1, 1.2, 1.5\}$ with one seed per point, keeping $\omega = 1$ as the linear baseline. The sparse variants (Section 6) set their window and gathered-segment lengths as reported where introduced.

Evaluation. Held-out cross-entropy is the primary metric throughout, evaluated once at the end of training on the fixed held-out sample above, at the training context length. Downstream accuracy (Section 7) is scored by log-likelihood on multiple-choice benchmarks with a standard evaluation harness. The positional-encoding extrapolation study (Section 3.2) evaluates each trained checkpoint at several context lengths beyond its training length, forward-only.