NEXUS: HIGHER-ORDER ATTENTION MECHANISMS IN TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers have achieved broad success across language and vision, using self-attention to model dependencies among input elements. Yet the relevance of one position to another often depends on evidence distributed across a wider context. Standard attention forms each matching score from a query and a key projected at the current layer, whose inputs may already carry context from earlier layers. Context-aware transformations and multi-token interactions enrich this process, motivating further study of how context can enter the matching operation itself. We introduce Nexus, a higher-order attention mechanism that refines queries and keys through separate causal attention branches before their final comparison. The construction extends recursively while reusing the original projections. Our analysis shows how other prefix tokens can influence comparisons within a single attention head. Experiments across four from-scratch model scales yield higher six-task averages than standard attention, and adapting pretrained Qwen2.5 models with Nexus improves reasoning-task averages after supervised fine-tuning. These results support recursive query/key refinement as a design for context-dependent matching in Transformer decoders.

1 INTRODUCTION

The Transformer architecture has become a foundation for sequence modeling, with successful applications in machine translation, language generation and visual recognition (Vaswani et al., 2017; Yang et al., 2024a; Li et al., 2023). Its ability to learn relationships among input elements provides a common basis for tasks with very different data and prediction targets. Central to this success is self-attention, which lets each position select and combine relevant information from its context. Direct connections between distant positions support long-range dependencies, while parallel computation makes the architecture well suited to large-scale training. These properties have motivated continued efforts to improve the attention mechanism itself.

In standard self-attention, the input states are projected into queries, keys and values. Query–key similarities determine the weights used to aggregate values, so each position receives a content-dependent summary of its visible context. Stacking layers allows these summaries to become progressively contextualized. Within an individual attention operation, however, each matching score is still computed directly from a projected query and key before further context is gathered into those matching states. This distinction matters when relevance depends on several tokens together: a phrase may resolve which entity an expression refers to, or an additional condition may determine which premise applies. Such examples motivate bringing distributed evidence into the comparison itself, so that the states used for matching can first incorporate the context available to each position.

Existing attention improvements offer several ways to introduce richer dependencies. Context-aware self-attention augments query/key transformations with contextual signals (Yang et al., 2019), while Multi-Token Attention combines information across query–key scores or weights (Golovneva et al., 2025). Explicit higher-order mechanisms model token tuples (Clift et al., 2020; Chakrabarti et al., 2026), and recurrent Transformers compose shared attention blocks through repeated hidden-state updates (Dehghani et al., 2019; Yang et al., 2024b). These approaches demonstrate the value of contextual representations and interactions beyond an isolated token pair. Their computational roles also differ: map-level operations refine comparisons after their scores are formed; full-block recurrence updates the hidden representation; and tuple-based operators change the objects being

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.

compared. Context-aware projections provide a closer starting point, raising the question of how to organize repeated context gathering on both sides of a match. We investigate whether causal attention can perform this refinement within the query and key branches while retaining a final aggregation of the original values.

We introduce **Nexus**, a higher-order attention mechanism that addresses this question through separate causal query and key refinement. Each branch gathers the context visible to its own position before the final query–key comparison. Nesting the same construction produces higher orders, with the original projections reused throughout. Our theoretical analysis establishes how prefix evidence changes matching at the level of a single attention head. Experiments across four from-scratch scales show higher six-task averages than Standard, while adapting pretrained Qwen2.5 models with Nexus improves reasoning-task averages after supervised fine-tuning. Together, these results connect a concrete higher-order construction with its matching properties and its use in newly trained and pretrained Transformers.

2 Related Work

Contextual and self-self state refinement. Context-aware self-attention injects global, deep or lower-layer context into query/key transformations (Yang et al., 2019); CoTNet contextualizes two-dimensional keys (Li et al., 2023); and KV Shifting blends keys and values with a one-token causal shift (Xu et al., 2025). CASTLE continually updates earlier keys as causal context grows (Song et al., 2026). GEM iterates and ensembles query–query, key–key and value–value self-self branches (Bousselham et al., 2024). Nexus forms separate causal query-side and key-side refinements, then brings their outputs together in a new cross-branch query–key comparison. Each refined key uses its own fixed causal prefix. This data path differs from both GEM's self-self map ensemble and CASTLE's continuing updates to earlier keys.

Map transformations, nesting and aggregation. MTA convolves query–key maps (Golovneva et al., 2025); SCLIP sums self-self maps (Wang et al., 2024); Talking-Heads and DCMHA mix maps across heads (Shazeer et al., 2020; Xiao et al., 2024). JAT updates both queries and keys with a shared graph derived from base cross-scores, yielding two-sided score propagation; higher orders use graph powers (Zhou et al., 2022). Nexus derives separate causal mixers from query–query and key–key similarities; recursive levels recompute them from refined states while retaining the original branch inputs as values. Luna nests latent packing and unpacking (Ma et al., 2021), while Compositional Attention selects among retrieval outputs (Mittal et al., 2022). Differential Transformer subtracts softmax maps, Attention on Attention gates aggregated features, and Tensor Product Attention factorizes per-token query/key/value states (Ye et al., 2025; Huang et al., 2019; Zhang et al., 2025). Appendix C compares these paths.

Polyadic and higher-order attention. The 2-Simplicial Transformer uses three-way attention and tensor products of values (Clift et al., 2020); tensor attention has been analyzed on Match3 (Sanford et al., 2023); and Strassen and Poly-Attention develop triplet and higher-order tensor constructions (Kozachinskiy et al., 2025; Chakrabarti et al., 2026). Nexus builds higher-order matching through nested pairwise attention: its order counts refinement levels leading to a final $n \times n$ map. This distinguishes recursive contextual matching from the explicit tuple scoring used by polyadic mechanisms.

Recurrence, sharing and sequence backbones. Universal Transformers recurrently apply a shared self-attentive transition, ALBERT shares encoder layers, and Perceiver repeatedly reads inputs through a latent bottleneck (Dehghani et al., 2019; Lan et al., 2020; Jaegle et al., 2021). Looped Transformers update a full hidden state with a shared block (Yang et al., 2024b), while Block-Recurrent Transformers carry state across sequence blocks (Hutchins et al., 2022). Other backbones replace or combine attention with recurrence and state-space components (Gu & Dao, 2024; De et al., 2024; Lieber et al., 2024; Ren et al., 2025; Dong et al., 2025). Nexus instead places recursion inside the projected query and key branches, while retaining the surrounding block and final value-aggregation interface.

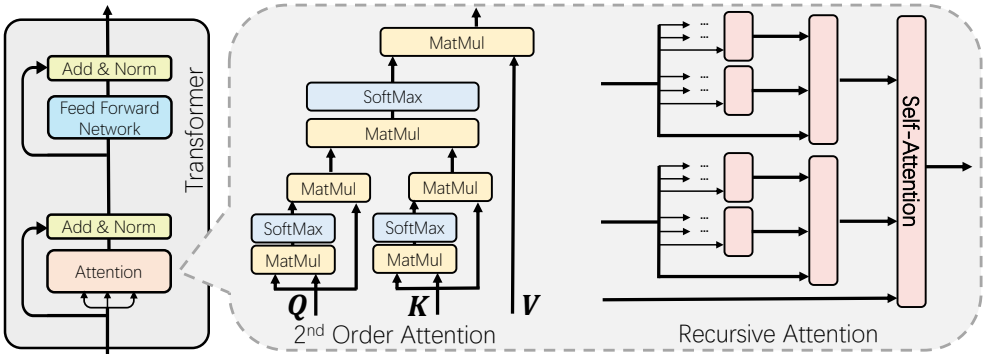


Figure 1: **Overview of Nexus.** Left: Nexus replaces the attention operator while preserving the residual and feed-forward paths. Middle: separate causal inner attentions refine the projected queries and keys before the final query–key attention aggregates V. Right: higher orders recursively refine the two branches. Equation 8 specifies the recursion, with the original branch inputs serving as values at every refinement level.

3 Nexus Attention

Nexus forms higher-order attention by contextualizing the states used for matching before aggregating values. We first define the causal attention operation, then construct its second-order and recursive forms.

Causal attention. Let $X \in \mathbb{R}^{n\times d}$ be a layer input, with token vectors written as rows. For one head of width d_h, standard attention forms

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

Let $M_{ij} = 0$ for $j \le i$ and $M_{ij} = -\infty$ otherwise. Define

$$\mathrm{Attn}_{M,\rho}(U, Z, Y) = \mathrm{softmax}\left(\frac{\rho(U)\rho(Z)^\top}{\sqrt{d_h}} + M\right) Y, \tag{2}$$

where softmax is row-wise. For fixed rotation matrices R_s, let $\rho_s(u) = uR_s^\top$ for a row vector u and $[\rho(U)]_s = \rho_s(U_s)$; without RoPE, ρ is the identity. Every inner and final operation uses the same causal mask. We call one score product, causal softmax and multiplication by the third argument an *attention core*; learned projections are counted separately. Equation 2 defines the deterministic operator, with training-time attention dropout specified in Appendix B.3.

3.1 From Single-Pair Scores to Higher-Order Matching

Standard attention constructs each score from token-wise queries and keys. Nexus first incorporates context into both matching states. Its second-order form is

$$B_Q(X) = \mathrm{softmax}\left(\frac{\rho(Q)\rho(Q)^\top}{\sqrt{d_h}} + M\right), \qquad \widehat{Q} = B_Q(X)Q, \tag{3}$$

$$B_K(X) = \mathrm{softmax}\left(\frac{\rho(K)\rho(K)^\top}{\sqrt{d_h}} + M\right), \qquad \widehat{K} = B_K(X)K, \tag{4}$$

$$\mathrm{Nexus}_2(X) = \mathrm{Attn}_{M,\rho}(\widehat{Q}, \widehat{K}, V) = A^N(X)V. \tag{5}$$

Here $A^N(X) \in \mathbb{R}^{n\times n}$ is the final causal softmax map, and $\mathrm{Nexus}_2(X) \in \mathbb{R}^{n\times d_h}$ is the head output. The final pre-scale score between positions i and j expands as

$$\rho_i(\widehat{Q}_i)\rho_j(\widehat{K}_j)^\top = \sum_{a\le i}\sum_{b\le j}(B_Q)_{ia}(B_K)_{jb}(Q_aR_i^\top)(K_bR_j^\top)^\top. \tag{6}$$

The indices have complementary roles. Positions i and j identify the query and candidate being compared, whereas $a \leq i$ and $b \leq j$ supply context through the two inner maps. Their context-dependent weights determine how the original projected states contribute to the final score. The output then aggregates V_j using A^N_{ij}: the prefix evidence changes the matching weights, while the original value branch supplies the retrieved content. This separates higher-order matching from final value aggregation. Appendix B.6 illustrates the inner and final maps on the 64-token example.

The two branches also have different causal contexts. Query refinement gathers evidence available at position i, while a refined key at position j summarizes $X_{\leq j}$. Evidence arriving between j and i can affect the comparison through the query branch; the previously formed key remains unchanged. In Equation 6, the mixed states are rotated at their destinations i and j. Accordingly, retrieved features are expressed in the destination coordinate frames, and each expanded interaction carries $R_i^\top R_j$.

Without positional rotations, let $S = QK^\top$. The complete final pre-scale score matrix factors as

$$\widehat{Q}\widehat{K}^\top = B_Q(QK^\top)B_K^\top = B_Q S B_K^\top. \tag{7}$$

The mixers are formed separately from the two projected spaces: B_Q depends on query–query similarities and B_K on key–key similarities. Thus the final matching rule uses both within-branch structure and cross-branch comparisons. Section 4 characterizes this dependence through pairwise log-odds and the information retained beyond the base cross-score.

Projection reuse. Nexus reuses the original query, key, value and output projections. The same query and key projections provide the states for inner refinement and final matching, and gradients train these roles jointly. During pretrained conversion, the projections retain their learned initialization while fine-tuning adapts the attention structure. The construction acts per head and preserves the original MHA/GQA grouping. Section 5.3 defines the branch-specific projections used by the unshared ablations; Appendix B gives pseudocode and the positional and head conventions.

3.2 Recursive Higher-Order Attention

The construction extends by nesting refinement inside both branches. Define $H^1(U, Z, Y) = \mathrm{Attn}_{M,\rho}(U, Z, Y)$ and, for $m \geq 2$,

$$H^m(U, Z, Y) = \mathrm{Attn}_{M,\rho}\big(H^{m-1}(U, U, U), H^{m-1}(Z, Z, Z), Y\big)\,. \tag{8}$$

Then $\mathrm{Nexus}_m(X) = H^m(XW_Q, XW_K, XW_V)$. An order-$m$ operator has $m-1$ branch-refinement levels followed by the final comparison: $m = 1$ is ordinary attention and $m = 2$ is Equation 5.

At each additional level, similarities are computed from previously refined states, and the resulting weights aggregate the original branch states. For example, the deterministic query-side states of the third-order construction are

$$Q^{(1)} = \mathrm{Attn}_{M,\rho}(Q, Q, Q), \qquad Q^{(2)} = \mathrm{Attn}_{M,\rho}(Q^{(1)}, Q^{(1)}, Q),$$

with the corresponding construction for K; the final operation is $\mathrm{Attn}_{M,\rho}(Q^{(2)}, K^{(2)}, V)$. All orders share the same causal support, and only the final map aggregates the original V. For fixed Q, K scaled by $\varepsilon \to 0$, retaining the original branch values gives each branch the same leading uniform-prefix average at every fixed positive refinement level. Proposition 3 quantifies the resulting local matching corrections across recursive orders.

Literal evaluation executes $2^m - 1$ attention cores. Deterministic, dropout-free evaluation permits sharing identical self-branch subexpressions, giving the $2m - 1$-core memoized schedule in Appendix B.1. The third-order experiment uses the literal seven-core schedule. All projection-reuse variants evaluate their projections, feed-forward layers and language-model head once; Section 5 and Appendix B.4 give the corresponding analytical forward costs.

Autoregressive evaluation. With attention dropout disabled, second-order Nexus appends the new raw states to Q, K, V, refines q_t, k_t from their causal histories, then appends $\widehat{k}_t$ and compares $\widehat{q}_t$ with $\widehat{K}_{\leq t}$ to aggregate $V_{\leq t}$. Stored states are unrotated; ρ is applied when forming logits. Each refined key uses only its own prefix and remains fixed during later steps. Appendix B.2 gives the explicit updates.

For H_Q query heads and H_{KV} key/value heads, this direct $Q, K, \widehat{K}, V$ cache contains $(H_Q + 3H_{KV})d_h$ persistent elements per layer and historical token, compared with $2H_{KV}d_h$ for a standard KV cache. The count excludes temporary activations and separately stored rotated copies. Other storage choices trade caching for recomputation.

The prototype composes standard attention calls rather than a dedicated fused kernel, so runtime also depends on kernel launches and intermediate memory traffic. Fusing the refinement stages and reducing persistent state are practical directions for implementing the operator.

4 Theoretical Analysis of Higher-Order Matching

We study the deterministic matching map of one causal attention head, treating X as the current-layer input variable. A score is *token-local* if it is determined by the query-token input and candidate-key-token input. Our analysis identifies how Nexus introduces dependence on other prefix tokens within this operator. Proofs appear in Appendix A.

4.1 Prefix-Conditioned Pairwise Preference

For visible candidates $j, \ell \le i$, define the pairwise log-odds

$$\Lambda_{i;j,\ell}(X) = \log \frac{A_{ij}(X)}{A_{i\ell}(X)}. \tag{9}$$

Theorem 1 (Separation from token-local scoring)**.** *Consider a causal attention head with finite visible logits $s_{ij}(X) = \phi_{ij}(x_i, x_j)$ and normalized row*

$$A^{\mathrm{local}}_{ij}(X) = \frac{\exp s_{ij}(X)}{\sum_{u \le i} \exp s_{iu}(X)}, \qquad j \le i,$$

where ϕ_{ij} may include arbitrary-width token-wise transformations and fixed positional information. Then

$$\Lambda^{\mathrm{local}}_{i;j,\ell}(X) = s_{ij}(X) - s_{i\ell}(X). \tag{10}$$

Changing a visible token x_r, with $r \le i$ and $r \notin \{i, j, \ell\}$, leaves these log-odds unchanged when x_i, x_j, x_ℓ remain fixed. There exists a second-order Nexus parameterization for which changing x_r changes these log-odds. The invariant also holds for a strictly positive token-local normalized kernel $A_{ij} = \kappa_{ij} / \sum_{u \le i} \kappa_{iu}$ by setting $s_{ij} = \log \kappa_{ij}$.

The token-local class includes linear or nonlinear token-wise projections, QK normalization, fixed RoPE and content-independent positional bias. Because the row normalizer cancels, widening a token-local score function does not remove this invariant.

Constructive example. Set $n = 4$, $d = d_h = 1$, $W_Q = W_K = 1$, use no positional rotation, and let $X(t) = (t, 0, 1, 1)^\top$ for $t \in \{0, 1\}$. At query $i = 4$, compare keys $j = 2$ and $\ell = 3$. Standard attention has $\Lambda^{\mathrm{std}}_{4;2,3} = -1$ on both inputs. Nexus gives approximately -0.421 and -0.307. Only x_1 changes, while the query-token and candidate-token inputs remain fixed. Thus Nexus changes their relative weighting through prefix evidence. Appendix A.2 gives the exact expressions.

4.2 Information Beyond the Base Query–Key Score

The factorization in Equation 7 suggests a second question: can the final map be recovered from the base cross-score alone?

Proposition 2 (Beyond the base query–key score)**.** *Let $S_\rho(Q, K) = [\rho_i(Q_i)\rho_j(K_j)^\top]_{ij}$ be the base position-rotated cross-score. Under the stated inner-source/final-destination RoPE convention, pairs $(Q_c, K_c) = (cQ, K/c)$ can have the same S_ρ but different normalized final causal Nexus maps. Therefore no deterministic function of S_ρ and the fixed mask alone equals the Nexus map for all Q, K.*

Reciprocal scaling preserves every base query–key score but scales the inner query–query logits by c^2 and the key–key logits by c^{-2}. The two refinement maps can therefore change while the

base cross-score stays fixed. The construction in Appendix A.3 shows that this changes the final normalized map as well. Nexus uses the separate similarity structures of the query and key spaces, not only their original cross-score.

4.3 Local Dependence on Recursive Order

For fixed finite Q, K, let $A_\varepsilon^{(m)}$ denote the final causal map in $H^m(\varepsilon Q, \varepsilon K, V)$ and $\Lambda_{i;j,\ell}^{(m)}(\varepsilon)$ its visible pairwise log-odds (Equation 9).

Proposition 3 (Local matching corrections)**.** *For the deterministic operator and recursion in Equations 2 and 8, with fixed sequence length, head width, causal mask and positional rotations, every fixed integer $m \geq 1$ satisfies, as $\varepsilon \to 0$,*

$$\Lambda_{i;j,\ell}^{(m+1)}(\varepsilon) - \Lambda_{i;j,\ell}^{(m)}(\varepsilon) = O(\varepsilon^{2m}),$$
$$A_\varepsilon^{(m+1)} - A_\varepsilon^{(m)} = O(\varepsilon^{2m}).$$

The map bound is entrywise. The maps and visible log-odds are analytic even functions of ε near zero. A single two-position input has a nonzero degree-$2m$ term in both differences for every $m \geq 1$.

Successive orders thus share all Taylor coefficients below degree $2m$. Using previous branch outputs as values can instead change the quadratic coefficient at each additional order. Appendix A.6 gives the proof and a finite-scale RoPE example that isolates this choice of branch values.

Let Z_m be the complete final score of $H^m(Q, K, V)$ before division by $\sqrt{d_h}$, masking and softmax. Without rotations, $\mathrm{rank}(Z_m) = \mathrm{rank}(QK^\top) \leq d_h$; fixed positional rotations preserve the bound d_h (Proposition 5).

5 Experiments

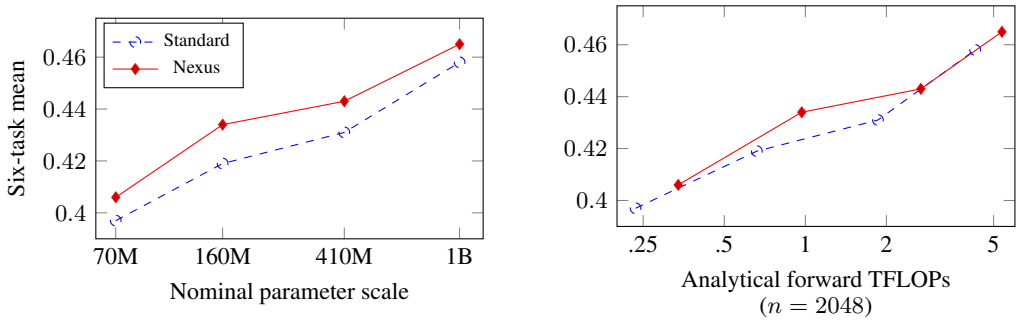
We evaluate higher-order attention through from-scratch pretraining, architectural ablations and pretrained-model conversion. The experiments examine performance across scales, the roles of the refinement branches and recursive order, and adaptation of existing projection spaces.

5.1 Setup

From-scratch pretraining. We train Pythia-style decoders (Biderman et al., 2023) on the Pile (Gao et al., 2020) for approximately 300B token positions per run at each scale. Standard/Nexus pairs use matched tokenizers, data and training configurations; Appendix B.5 specifies the 143,000-update budget and scale-specific optimization settings. We evaluate zero-shot accuracy on ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020) and SciQ (Welbl et al., 2017). Avg. is their unweighted macro-average. Nexus denotes the second-order projection-reuse variant unless otherwise specified.

Training runs and controls. Results are means over three independent training runs per condition, each a complete pretraining or SFT rerun. Each resulting checkpoint is scored separately before averaging. At 70M, we compare refinement branches, projection reuse and recursive order, together with a three-call hidden-state loop (Yang et al., 2024b), an 11-layer Standard model, Differential Transformer (Ye et al., 2025), MTA (Golovneva et al., 2025) and a Nexus–Diff hybrid. The 11-layer model is the nearest integer-depth match to six-layer Nexus under the analytical forward-cost convention at context length 2048. Section 5.3 defines the refinement branches; Appendix B.3 specifies the hidden-state loop and Nexus–Diff.

Analytical costs. Forward FLOPs count dense matrix multiplications in attention, projections, feed-forward layers and the language-model head, with two FLOPs per multiply–accumulate. The main comparison uses $n = 2048$. With the shared Pythia training workload and a common multiplier from forward to forward-plus-backward work, these ratios also define an analytical dense-matmul training-compute proxy. Appendix B.4 gives the accounting, assumptions and $n = 512$ values. Runtime and cached decoding remain distinct from this arithmetic count.



(a) Accuracy versus parameters

(b) Forward cost at $n = 2048$

Figure 2: **Pythia-style scaling.** Six-task accuracy versus nominal parameter scale (left) and analytical forward FLOPs for a 2048-token sequence (right). Both horizontal axes are logarithmic. The 160M Nexus model reaches 0.434 at 0.970 TFLOPs, compared with 0.431 at 1.860 TFLOPs for 410M Standard. Table 1 gives task-level results, and Appendix Table 7 gives the 512-token cost accounting.

Table 1: Task-level zero-shot accuracy. Avg. is the unweighted six-task mean; bold marks the higher displayed value within each scale. HSwag denotes HellaSwag.

Scale	Model	ARC-C	ARC-E	HSwag	LogiQA	PIQA	SciQ	Avg.
70M	Standard	**0.208**	**0.359**	0.356	**0.277**	0.569	0.615	0.397
70M	Nexus	0.206	0.357	**0.367**	0.273	**0.601**	**0.630**	**0.406**
160M	Standard	0.200	0.385	0.380	0.260	0.600	0.686	0.419
160M	Nexus	**0.211**	**0.405**	**0.385**	**0.285**	**0.605**	**0.713**	**0.434**
410M	Standard	0.225	0.394	0.375	0.285	0.601	0.708	0.431
410M	Nexus	**0.226**	**0.415**	**0.384**	**0.294**	**0.608**	**0.733**	**0.443**
1B	Standard	**0.232**	0.440	0.395	**0.296**	0.625	0.758	0.458
1B	Nexus	0.230	**0.455**	**0.399**	0.290	**0.636**	**0.777**	**0.465**

5.2 Scaling from Scratch

Nexus improves the six-task average over Standard by 0.009, 0.015, 0.012 and 0.007 at 70M, 160M, 410M and 1B, respectively. It improves all six tasks at 160M and 410M, and three and four at 70M and 1B (Table 1). At each scale, all nine cross-run macro comparisons favor Nexus (Appendix Table 11). Nexus also has lower mean next-token NLL on the shared evaluation sample at each scale (Appendix Table 9).

Across model sizes, 160M Nexus reaches 0.434, compared with 0.431 for 410M Standard. At sequence length 2048, their analytical forward costs are 0.970 and 1.860 TFLOPs. The smaller Nexus model uses 61% fewer nominal parameters and approximately 48% fewer forward FLOPs under this accounting. Figure 2 shows the accuracy–size and accuracy–cost comparisons. Appendix Table 8 also gives the 512-token costs; their different ratio follows from the quadratic sequence-length term in attention.

5.3 Ablations and Compute Accounting

The 70M ablations compare refinement branches, projection reuse and recursive order (Table 2); Appendix Table 5 gives task-level scores.

Refinement branches. With Q, K, V from Equation 1, each refinement branch $r \in \{q, k, v\}$ uses an independent projection triplet:

$$R_r(X) = \mathrm{Attn}_{M,\rho}(XW_Q^{(r)}, XW_K^{(r)}, XW_V^{(r)}). \tag{11}$$

Table 2: 70M design ablations. Branch triplets count independent Q/K/V projection triplets used by the refinement branches. Cores count attention calls; bold marks the highest average.

Method	States	Order/loops	Branch triplets	Cores	Avg.
Standard	-	-	0	1	0.397
Nexus-Q	Q	2	1	2	0.400
Nexus-QK	Q,K	2	2	3	0.409
Nexus-QKV	Q,K,V	2	3	4	0.409
Nexus (proj. reuse)	Q,K	2	0	3	0.406
Nexus–Recursive	Q,K	3	0	7	**0.415**
Hidden-state loop	Hidden	`2 loops`	0	3	0.402

Branch-triplet counts describe the paths in Equation 11, rather than net parameter increments. The hidden-state loop performs three complete attention-sublayer updates, with `2 loops` denoting two additional calls. Section 5.3 defines the refinement branches; Appendix B.3 specifies the hidden-state loop.

The three variants are

$$\text{Nexus-Q}(X) = \text{Attn}_{M,\rho}(R_q(X), K, V), \tag{12}$$

$$\text{Nexus-QK}(X) = \text{Attn}_{M,\rho}(R_q(X), R_k(X), V), \tag{13}$$

$$\text{Nexus-QKV}(X) = \text{Attn}_{M,\rho}(R_q(X), R_k(X), R_v(X)). \tag{14}$$

They use one, two and three independent branch triplets and two, three and four attention cores, respectively. Default Nexus reuses the original projections in Equations 3 and 4.

Among the branch-specific configurations, Nexus-Q reaches 0.400 and Nexus-QK reaches 0.409. Nexus-QKV also reaches 0.409, using one more branch and attention core than Nexus-QK. The query/key configuration therefore attains the highest second-order average in this group while retaining the original V branch for final aggregation.

Projection reuse. Reusing the original projections gives 0.406, compared with 0.409 for branch-specific query/key projections. This retains most of the latter configuration's gain over Standard while introducing no additional projection matrices. The three-core projection-reuse construction defines default Nexus.

Recursive order. Third-order Nexus–Recursive reaches 0.415, the highest average in the 70M design ablations. It adds another query/key-refinement level and uses the literal seven-core execution, costing approximately $2.32\times$ Standard at $n = 2048$. The five-core deterministic memoization is described separately in Appendix B.1; the experimental row uses the seven-core count.

Architectural controls. The three-call hidden-state loop reaches 0.402. The 11-layer Standard model reaches 0.403 at $1.46\times$ the six-layer Standard forward cost, while default Nexus reaches 0.406 at $1.44\times$ (Table 3). Nexus also has lower mean NLL than these controls on the shared held-out sample: 2.8508, compared with 2.8668 for 11-layer Standard and 2.8760 for the hidden-state loop (Appendix Table 10). Seven of nine cross-run macro comparisons favor Nexus over 11-layer Standard (Appendix Table 11). The depth control adds complete standard layers, and the hidden-state loop repeats normalization, projections and residual updates; Nexus instead refines the projected matching states.

5.4 Attention Controls and Pretrained Retrofit

Attention controls. At 70M, Nexus and the reported MTA control both reach 0.406. MTA uses $1.07\times$ Standard's analytical forward compute, compared with $1.44\times$ for Nexus under the instantiated layouts (Table 3; Appendix B.4). Nexus has lower mean NLL, 2.8508 versus 2.8683 (Appendix Table 10), while five of nine cross-run macro comparisons favor Nexus (Appendix Table 11). This is a lower-NLL, higher-compute tradeoff, with the same displayed six-task average. Combining query/key refinement with a Differential final map gives 0.410, compared with 0.406 for Nexus and 0.401 for Differential attention alone. Nexus–Diff retains default Nexus's $1.44\times$ analytical dense-matmul forward cost at $n = 2048$.

Table 3: Attention variants and architectural controls. Relative analytical forward FLOPs are for $n = 2048$, normalized to six-layer Standard. The attention variants use the nominal 70M scale; the 11-layer control adds standard layers and therefore changes parameter count.

Method	Relative FLOPs ($n = 2048$)	Avg.
Standard 6 layers	1.00×	0.397
Diff. Transformer	—	0.401
MTA (6L control)	1.07×	0.406
Nexus (proj. reuse)	1.44×	0.406
Nexus–Diff	1.44×	0.410
Standard 11 layers	1.46×	0.403
Hidden-state loop	1.66×	0.402
Nexus–Recursive	2.32×	**0.415**

The MTA-control entry uses the six-layer configuration specified in Appendix B.3; its $L/4$ rate-normalized reference is $1.08\times$ (Equation 48). The standalone Differential Transformer row is the accuracy baseline for Nexus–Diff. Appendix B.4 gives all cost formulas.

Table 4: Qwen2.5 reasoning accuracy after SFT. Bold marks the higher displayed result or a tie within each base model. Avg. is the unweighted task mean.

Base	Architecture	MATH-500	AIME24	GPQA-D.	Avg.
1.5B	Standard	0.786	**0.194**	0.276	0.419
1.5B	Nexus	**0.801**	**0.194**	**0.280**	**0.425**
7B	Standard	**0.921**	0.452	0.401	0.591
7B	Nexus	**0.921**	**0.475**	**0.407**	**0.601**

Each checkpoint score averages correctness over 16 stochastic decodes per problem; table entries then average the three independent training runs. Equation 49 gives the aggregation.

Pretrained-model conversion. We replace attention in Qwen2.5-Base at 1.5B and 7B (Yang et al., 2024a) with projection-reuse Nexus and perform response-only supervised fine-tuning using Open-R1 (Hugging Face, 2025). Each pair starts from the same base checkpoint and uses matched data, SFT configurations and decoding. Table 4 reports accuracy on MATH-500 (Lightman et al., 2023; Hugging Face H4, 2024), AIME24 (Hugging Face H4, 2025) and GPQA-Diamond (Rein et al., 2024). For each checkpoint, task accuracy averages correctness over 16 stochastic decodes per problem; the resulting scores are then averaged over the three training runs. Appendix B.5 records the stated SFT conditions, decoding settings and aggregation formula.

The 1.5B average increases from 0.419 to 0.425, with gains on MATH-500 and GPQA-Diamond and a tie on AIME24 at the displayed precision. The 7B average increases from 0.591 to 0.601, with gains on AIME24 and GPQA-Diamond and a tie on MATH-500. At each scale, all nine cross-run comparisons of the three-task macro-average favor Nexus over Standard (Appendix Table 11).

6 Conclusion

Nexus constructs higher-order attention by recursively refining queries and keys before their final comparison. Separate causal branches let prefix evidence shape the matching states, while the final map aggregates the original values. The construction reuses the original projections and fits within the existing Transformer block. Our analysis establishes a per-head separation from token-local scoring, identifies information beyond the base query–key score, and characterizes local matching corrections across recursive orders. Default Nexus achieves higher six-task averages across four from-scratch scales; the 70M study supports joint query/key refinement and attains its highest design-ablation average with the third-order variant. Pretrained Qwen2.5 conversion also improves reasoning-task averages after supervised fine-tuning. Together, these findings identify recursive query/key refinement as a concrete higher-order attention design for newly trained and pretrained Transformers.

AI USE STATEMENT

Generative AI assisted with manuscript reconstruction from earlier source and PDF versions; literature discovery and synthesis; drafting and editing; mathematical claims, proofs and consistency checks; interpretation of experimental materials; and preparation of tables and references. AI assistance during the development of this work also included operator implementation and testing, and the design and auditing of experimental workflows. Experimental results are reported from the existing measurements; analytical quantities follow from the stated equations and architecture dimensions. The authors take responsibility for the complete submission, including its text, code, equations, proofs, claims, citations and numerical results.

REPRODUCIBILITY STATEMENT

Section 3 specifies the causal attention operator, projection reuse and recursive construction. Appendix A provides proofs and explicit constructions. Appendix B gives pseudocode, architectural variants, RoPE/GQA conventions, incremental equations, analytical FLOP formulas and evaluation settings. Results are means over three independent end-to-end training runs. The tables provide scale comparisons, task-level design ablations, attention controls and pretrained-model results. Within-pair comparisons use matched protocols based on Pythia/Pile and Qwen/Open-R1. Appendix B.5 specifies Qwen's stochastic decoding and score aggregation, together with the cited repository and dataset revisions.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 2397–2430, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Walid Bousselham, Felix Petersen, Vittorio Ferrari, and Hilde Kuehne. Grounding everything: Emerging localization properties in vision-language transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3828–3837, 2024.

Sayak Chakrabarti, Toniann Pitassi, and Josh Alman. Poly-attention: A general scheme for higher-order self-attention. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=amivrmQyvQ.

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. arXiv:1803.05457, 2018.

James Clift, Dmitry Doryn, Daniel Murfet, and James Wallbridge. Logic and the 2-simplicial transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkecJ6VFvr.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando de Freitas, and Caglar

Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *International Conference on Learning Representations*, pp. 97729–97755, 2025.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Olga Golovneva, Tianlu Wang, Jason E. Weston, and Sainbayar Sukhbaatar. Multi-token attention. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.00927v2.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4634–4643, 2019.

Hugging Face. Open-R1: A fully open reproduction of deepseek-r1. GitHub repository, 2025. URL https://github.com/huggingface/open-r1/tree/cec57f3a550a426801985ee15535e7fd459aafd2. Git revision cec57f3a550a426801985ee15535e7fd459aafd2.

Hugging Face H4. MATH-500. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/MATH-500/tree/2343f79f3640c0f795bbf4e3234396cfe9266d0f. Default configuration, test split; revision 2343f79f3640c0f795bbf4e3234396cfe9266d0f.

Hugging Face H4. AIME 2024. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024/tree/58e32b4b387f34192acdacbdf602510c459a6525. Default configuration, train split; revision 58e32b4b387f34192acdacbdf602510c459a6525.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 33248–33261, 2022. URL https://openreview.net/forum?id=uloenYmLCAo.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4651–4664, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Alexander Kozachinskiy, Felipe Urrutia, Hector Orellana, Tomasz Steifer, German Pizarro, Matías Fuentes, Francisco Meza Vasquez, Cristian Buc Calderon, and Cristobal Rojas. Strassen attention, split VC dimension and compositionality in transformers. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1489–1500, 2023. doi: 10.1109/TPAMI.2022.3164083.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. arXiv:2007.08124, 2020.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2441–2453, 2021.

Sarthak Mittal, Sharath Chandra Raparthy, Irina Rish, Yoshua Bengio, and Guillaume Lajoie. Compositional attention: Disentangling search and retrieval. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=IwJPj2MBcIa.

Tam Nguyen, Tan M. Nguyen, and Richard G. Baraniuk. Mitigating over-smoothing in transformers via regularized nonlocal functionals. arXiv:2312.00751, 2023. URL https://arxiv.org/abs/2312.00751.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations*, pp. 53551–53575, 2025.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou. Talking-heads attention, 2020.

Zhuoqing Song, Peng Sun, Huizhuo Yuan, and Quanquan Gu. Causal attention with lookahead keys. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OGQJsUUtEB.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2021. URL https://arxiv.org/abs/2104.09864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Feng Wang, Jieru Mei, and Alan Yuille. SCLIP: Rethinking self-attention for dense vision-language inference. In *Computer Vision – ECCV 2024*, volume 15079, pp. 315–332, 2024. doi: 10.1007/978-3-031-72664-4_18.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. arXiv:1707.06209, 2017.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Improving transformers with dynamically composable multi-head attention. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 54300–54318, 2024. URL https://proceedings.mlr.press/v235/xiao24d.html.

Mingyu Xu, Bingning Wang, and Weipeng Chen. KV shifting attention enhances language modeling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 69725–69748, 2025. URL https://proceedings.mlr.press/v267/xu25ac.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024a. URL https://arxiv.org/abs/2412.15115.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. Context-aware self-attention networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 387–394, 2019. doi: 10.1609/aaai.v33i01.3301387.

Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=HHbRxoDTxE.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. In *International Conference on Learning Representations*, pp. 144–164, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? arXiv:1905.07830, 2019.

Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Zhen Qin, Yang Yuan, Quanquan Gu, and Andrew Yao. Tensor product attention is all you need. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Haoyi Zhou, Siyang Xiao, Shanghang Zhang, Jieqi Peng, Shuai Zhang, and Jianxin Li. Jump self-attention: Capturing high-order statistics in transformers. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. Value residual learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28341–28356, 2025. doi: 10.18653/v1/2025.acl-long.1375. URL https://aclanthology.org/2025.acl-long.1375/.

A Additional Theory and Proofs

All results in this appendix concern the deterministic attention map in Equation 2, with finite logits at visible positions.

A.1 Linear-Projection Geometry

Proposition 4 (Visible-prefix subspace)**.** *Consider one standard causal-attention head with token-wise linear query/key projections and no position-dependent rotations. For its strictly positive visible-prefix distribution $a_i = (A_{i1}, \dots, A_{ii})^\top$, there are $u_i \in \mathbb{R}^d$ and $c_i \in \mathbb{R}$ such that*

$$\log a_i = X_{\le i} u_i + c_i \mathbf{1}_i. \tag{15}$$

Thus, conditional on X, $\log a_i \in \mathrm{Col}(X_{\le i}) + \mathrm{span}(\mathbf{1}_i)$.

Proof. The active logits at query i satisfy

$$z_i = \frac{K_{\le i} q_i^\top}{\sqrt{d_h}} = X_{\le i} \underbrace{\frac{W_K q_i^\top}{\sqrt{d_h}}}_{u_i}. \tag{16}$$

Since $a_i = \mathrm{softmax}(z_i)$,

$$\log a_i = z_i - \log\left(\sum_{s \le i} \exp z_{is}\right) \mathbf{1}_i = X_{\le i} u_i + c_i \mathbf{1}_i. \tag{17}$$

Restricting to the visible prefix avoids logarithms of mask-induced zeros. □

For fixed projected keys, the same calculation gives the tighter inclusion

$$\log a_i \in \mathrm{Col}(K_{\le i}) + \mathrm{span}(\mathbf{1}_i), \qquad \dim(\mathrm{Col}(K_{\le i}) + \mathrm{span}(\mathbf{1}_i)) \le \min(i, d_h + 1).$$

The head-width bound remains informative when the input-space bound spans all of $\mathbb{R}^i$, for example at a full-row-rank prefix with $d_h + 1 < i \le d$. With RoPE, define the rotated key matrix $\widetilde{K} = \rho(K)$. Then

$$\log a_i \in \mathrm{Col}(\widetilde{K}_{\le i}) + \mathrm{span}(\mathbf{1}_i), \qquad \dim\Big(\mathrm{Col}(\widetilde{K}_{\le i}) + \mathrm{span}(\mathbf{1}_i)\Big) \le \min(i, d_h + 1).$$

Thus fixed positional rotations preserve a projected-key dimension bound, with the relevant column space defined by the rotated rather than raw keys.

A.2 Proof of Theorem 1

For $j, \ell \le i$, the mask entries are zero and the row-wise normalizer cancels:

$$\log \frac{A_{ij}}{A_{i\ell}} = \left(s_{ij} - \log \sum_{u \le i} e^{s_{iu}}\right) - \left(s_{i\ell} - \log \sum_{u \le i} e^{s_{iu}}\right) \tag{18}$$

$$= s_{ij} - s_{i\ell} = \phi_{ij}(x_i, x_j) - \phi_{i\ell}(x_i, x_\ell). \tag{19}$$

Changing a visible x_r with $r \le i$ and $r \notin \{i, j, \ell\}$ therefore leaves the ratio unchanged when x_i, x_j, x_ℓ remain fixed. Standard row-vector attention with fixed rotations is the special case

$$s_{ij} = \frac{(x_i W_Q R_i^\top)(x_j W_K R_j^\top)^\top}{\sqrt{d_h}}.$$

QK normalization and fixed positional bias can be absorbed into ϕ_{ij}. For a strictly positive token-local normalized kernel, use $s_{ij} = \log \kappa_{ij}$.

For the Nexus example $X(t) = (t, 0, 1, 1)^\top$, inner attention is an exponentially weighted scalar average. At $t = 0$,

$$\widehat{q}_4 = \frac{e}{1+e}, \qquad \widehat{k}_2 = 0, \qquad \widehat{k}_3 = \frac{e}{2+e}, \tag{20}$$

which gives

$$\Lambda^{\mathrm{Nexus}}_{4;2,3}(X(0)) = -\frac{e^2}{(1+e)(2+e)} \approx -0.421. \tag{21}$$

At $t = 1$,

$$\widehat{q}_4 = \frac{3e}{1+3e}, \qquad \widehat{k}_2 = \frac{1}{2}, \qquad \widehat{k}_3 = \frac{2e}{1+2e}, \tag{22}$$

and hence

$$\Lambda^{\mathrm{Nexus}}_{4;2,3}(X(1)) = \frac{3e(1-2e)}{2(1+3e)(1+2e)} \approx -0.307. \tag{23}$$

The two inputs differ only at $r = 1$ relative to the fixed triple $(i, j, \ell) = (4, 2, 3)$, yet their log-odds differ. No token-local single-head scoring map agrees with this Nexus map on both inputs.

A.3 Proof of Proposition 2

Take $n = d_h = 2$, using sequence indices $1, 2$ for rotary positions $0, 1$. Let the rotations be I and

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad \theta = 1.$$

For $c > 0$, define

$$Q_c = c\begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}, \qquad K_c = c^{-1}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{24}$$

The complete base RoPE cross-score is

$$S_\rho(Q_c, K_c) = \begin{pmatrix} 0 & -\cos\theta \\ -\sin\theta & 1 \end{pmatrix}, \tag{25}$$

independent of c. Under the causal mask, write

$$p_c = \left(1 + e^{c^2(1+\cos\theta)/\sqrt{2}}\right)^{-1}, \qquad q_c = \left(1 + e^{(1+\sin\theta)/(c^2\sqrt{2})}\right)^{-1}. \tag{26}$$

The inner-source/final-destination convention gives $\widehat{q}_2 = c(1 - 2p_c, 0)$, $\widehat{k}_1 = c^{-1}(0, -1)$ and $\widehat{k}_2 = c^{-1}(1 - q_c, -q_c)$. Its final row-2 log-odds are

$$\Lambda_c = \log\frac{A^N_{21}}{A^N_{22}} = -\frac{(1-2p_c)(1+\sin\theta - q_c)}{\sqrt{2}}. \tag{27}$$

The dependence on c follows directly from the limits

$$\lim_{c\downarrow 0}\Lambda_c = 0, \qquad \lim_{c\to\infty}\Lambda_c = -\frac{\frac{1}{2} + \sin 1}{\sqrt{2}} < 0.$$

Thus Λ_c is not constant even though the complete base cross-score is fixed. The expressions also give $\Lambda_1 \in (-0.572, -0.571)$ and $\Lambda_2 \in (-0.981, -0.980)$. Identical base cross-scores therefore produce distinct final Nexus maps. Scalar scaling commutes with the fixed rotations, which is the symmetry used in this construction.

Inner similarity scales. The reciprocal scaling preserves the cross-score while multiplying the two inner self-similarity scores by c^2 and c^{-2}. It consequently changes their softmax concentration in opposite directions. This observation concerns the unnormalized projected states in the defined operator; projected-state normalization changes the scale symmetry.

A separate raw-score observation. In an unrotated head, take

$$Q = \begin{pmatrix} -4 & -3 \\ -2 & -2 \end{pmatrix}, \qquad K = Q^{-\top} = \begin{pmatrix} -1 & 1 \\ 3/2 & -2 \end{pmatrix}, \tag{28}$$

so $S = I_2$. The causal mixers are

$$B_Q = \begin{pmatrix} 1 & 0 \\ u & 1-u \end{pmatrix}, \qquad B_K = \begin{pmatrix} 1 & 0 \\ v & 1-v \end{pmatrix}, \quad u = \sigma(3\sqrt{2}), \quad v = \sigma(-39\sqrt{2}/8),$$

where $\sigma(z) = (1 + e^{-z})^{-1}$. The visible entries of $T = B_Q B_K^\top$ satisfy

$$T_{11} = 1, \qquad T_{21} = u, \qquad T_{22} = w := uv + (1-u)(1-v). \tag{29}$$

A shared propagation $GSG^\top = GG^\top$ matching these entries would, by symmetry, equal $C = \left(\begin{smallmatrix} 1 & u \\ u & w \end{smallmatrix}\right)$. But $u > 0.9$ and $v < 0.1$, so $w < v + (1-u) < 0.2 < u^2$. Hence $\det C = w - u^2 < 0$, contradicting $GG^\top \succeq 0$ for any real G. This obstruction concerns raw scores. Row-wise softmax is invariant to row shifts, so the normalized-map question is distinct and is resolved by the scalar-gauge construction above.

Direct scalar fusion. For $n = 2$, $d_h = 1$, no rotations and $Q = K = (0, 1)^\top$, let $p = (1 + e)^{-1}$ and $r = e/(1 + e)$. Then

$$B_Q = B_K = B = \begin{pmatrix} 1 & 0 \\ p & r \end{pmatrix}. \tag{30}$$

Every convex fusion $D_\lambda = \lambda B_Q + (1 - \lambda) B_K$ equals B, while

$$QK^\top = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (B_Q Q)(B_K K)^\top = B(QK^\top)B^\top = \begin{pmatrix} 0 & 0 \\ 0 & r^2 \end{pmatrix}. \tag{31}$$

The final Nexus map has $(A^N)_{21} = (1 + e^{r^2})^{-1} > (1 + e)^{-1} = B_{21}$ because $0 < r^2 < 1$. If an unnormalized scalar fusion $\alpha B_Q + \beta B_K$ equaled A^N, its first row would force $\alpha + \beta = 1$, yielding the same contradiction. This compares direct scalar fusion of these two maps; Appendix C describes the corresponding data paths in related architectures.

A.4 Raw-Score Rank

Proposition 5 (Rank conservation without rotations)**.** *For finite visible logits and no position-dependent rotations, the complete order-m pre-scale, pre-mask score satisfies*

$$Z_m = P_Q^{(m-1)} S (P_K^{(m-1)})^\top, \qquad m \geq 1,$$

where $P_Q^{(0)} = P_K^{(0)} = I_n$ and, for $m \geq 2$, the two $P_Q^{(m-1)}$ and $P_K^{(m-1)}$ are invertible lower-triangular row-stochastic causal-softmax mixers. Hence $\operatorname{rank}(Z_m) = \operatorname{rank}(S) \leq d_h$. With fixed position-dependent rotations, the complete final raw score has rank at most d_h.

Proof. Every causal-softmax matrix with finite visible logits is lower triangular, row stochastic and has a strictly positive diagonal, hence is invertible. For $m \geq 2$, the final query-side branch uses the original Q as values, so it equals one such mixer times Q; the same holds for K. Equation 7 gives the factorization and rank equality. With rotations, the refined-and-rotated state matrices each have d_h columns, so their product has rank at most d_h. □

The proposition concerns the complete finite score matrix before masking and softmax. Normalization produces the attention map used for value aggregation.

A.5 Causality and Head Structure

All inner and final operations use the same causal mask, so a refined state at position i depends only on $X_{\leq i}$. Inner scores rotate base states at their source positions; the final comparison rotates mixed states at their destination positions. Theorem 1 applies to each standard attention head, including each GQA query head. Value aggregation, concatenation and output projection subsequently combine the head outputs.

A.6 Proof of Proposition 3

All expansions below hold for fixed finite input matrices, head width, positions and recursion order. The positional map ρ is fixed and linear, as in Equation 2, and attention is deterministic. Analyticity is considered on visible entries; masked entries are identically zero.

Proof. **Scaled branch recursion.** For a branch input U, recall

$$T_0(U) = U, \qquad T_{r+1}(U) = \mathrm{Attn}_{M,\rho}(T_r(U), T_r(U), U).$$

Introduce the scalar s and the maps

$$\Phi_s(Z) = \mathrm{softmax}\left(\frac{s\,\rho(Z)\rho(Z)^\top}{\sqrt{d_h}} + M\right), \qquad \mathcal{F}_U(s, Z) = \Phi_s(Z)U.$$

Define

$$Z_0(s; U) = U, \qquad Z_{r+1}(s; U) = \mathcal{F}_U(s, Z_r(s; U)).$$

Linearity of ρ and induction give

$$T_r(\varepsilon U) = \varepsilon Z_r(\varepsilon^2; U).$$

At $s = 0$, every visible row is uniform. Writing

$$C_{ij} = \begin{cases} 1/i, & j \le i, \\ 0, & j > i, \end{cases}$$

we have $\mathcal{F}_U(0, Z) = CU$ independently of Z. The functions $Z_r(s; U)$ are analytic near zero, and $Z_r(0; U) = CU$ for every $r \ge 1$.

The derivative of the visible score matrix in a direction E is

$$\frac{s}{\sqrt{d_h}}\left(\rho(E)\rho(Z)^\top + \rho(Z)\rho(E)^\top\right).$$

The softmax derivative is analytic, so on a fixed bounded convex neighborhood of the relevant states, the derivative of $\mathcal{F}_U$ with respect to Z has norm $O(|s|)$. The mean-value formula therefore gives

$$\mathcal{F}_U(s, Z) - \mathcal{F}_U(s, W) = O(|s|\,\|Z - W\|)$$

for states in that neighborhood.

Starting from $Z_1 - Z_0 = O(1)$ and applying this bound inductively yields

$$Z_{r+1}(s; U) - Z_r(s; U) = O(s^r), \qquad r \ge 0.$$

Consequently,

$$T_{r+1}(\varepsilon U) - T_r(\varepsilon U) = O(\varepsilon^{2r+1}), \qquad T_r(\varepsilon U) = O(\varepsilon).$$

Final normalized matching. The order-m final logits are formed from $T_{m-1}(\varepsilon Q)$ and $T_{m-1}(\varepsilon K)$. Increasing the order changes either state by $O(\varepsilon^{2m-1})$, while both states are $O(\varepsilon)$. Bilinearity of the final score therefore gives an $O(\varepsilon^{2m})$ change in every logit. For visible $j, \ell \le i$, the row normalizer cancels in the log-odds, which proves the stated bound.

The branch representation $T_r(\varepsilon U) = \varepsilon Z_r(\varepsilon^2; U)$ also shows that the final logits, normalized maps and visible log-odds are analytic functions of ε^2 near zero. Thus their Taylor expansions contain only even powers. Smoothness of row-wise softmax additionally gives

$$A_\varepsilon^{(m+1)} - A_\varepsilon^{(m)} = O(\varepsilon^{2m})$$

entrywise. Constants in these bounds may depend on the fixed inputs, head width, positions and order.

Sharpness for the same input at every order. Take $n = 2$, $d_h = 1$, no positional rotation, and $Q = K = (0, 1)^\top$. This is realized by the one-dimensional projections $W_Q = W_K = 1$ on $X = (0, 1)^\top$. Let $\sigma(z) = (1 + e^{-z})^{-1}$ and set $s = \varepsilon^2$. The first branch state is always zero, and the second is $\varepsilon a_r(s)$, where

$$a_0(s) = 1, \qquad a_{r+1}(s) = \sigma\left(s\,a_r(s)^2\right).$$

The final row-2 log-odds comparing position 2 with position 1 are

$$\Lambda_m(\varepsilon) := \Lambda_{2;2,1}^{(m)}(\varepsilon) = s\,a_{m-1}(s)^2.$$

Since $\sigma(z) = \frac{1}{2} + \frac{1}{4}z + O(z^3)$,

$$a_1(s) = \frac{1}{2} + \frac{1}{4}s + O(s^3), \qquad a_2(s) = \frac{1}{2} + \frac{1}{16}s + O(s^2).$$

Hence $a_2 - a_1 = -3s/16 + O(s^2)$. For $r \geq 1$, put $d_r = a_{r+1} - a_r$. Both a_r and a_{r+1} equal $1/2 + O(s)$, and the mean-value formula for σ gives

$$d_{r+1} = \Big(\frac{s}{4} + O(s^2)\Big)\, d_r.$$

Induction therefore yields

$$d_r = -\frac{3}{4^{r+1}}s^r + O(s^{r+1}), \qquad r \geq 1.$$

For $m \geq 2$,

$$\Lambda_{m+1} - \Lambda_m = s(a_m + a_{m-1})(a_m - a_{m-1}) = -\frac{3}{4^m}s^m + O(s^{m+1}).$$

For $m = 1$, direct substitution gives $s(a_1^2 - 1) = -3s/4 + O(s^2)$. Thus, for every $m \geq 1$,

$$\Lambda_{m+1}(\varepsilon) - \Lambda_m(\varepsilon) = -\frac{3}{4^m}\varepsilon^{2m} + O(\varepsilon^{2m+2}).$$

The coefficient is nonzero, proving sharpness. In this example $(A_\varepsilon^{(m)})_{22} = a_m(s)$, hence

$$(A_\varepsilon^{(m+1)} - A_\varepsilon^{(m)})_{22} = -\frac{3}{4^{m+1}}\varepsilon^{2m} + O(\varepsilon^{2m+2}).$$

This proves sharpness for the normalized map itself.

Effect of changing the branch values. To isolate the role of the third argument, consider instead

$$\widetilde{T}_0(U) = U, \qquad \widetilde{T}_{r+1}(U) = \mathrm{Attn}_{M,\rho}(\widetilde{T}_r(U), \widetilde{T}_r(U), \widetilde{T}_r(U)).$$

For each fixed r, expansion at zero gives

$$\widetilde{T}_r(\varepsilon U) = \varepsilon C^r U + O(\varepsilon^3).$$

Indeed, the claim holds at $r = 0$; if it holds at r, the next softmax map is $C + O(\varepsilon^2)$, which proves the induction step. By contrast, original-value recursion has

$$T_r(\varepsilon U) = \varepsilon C U + O(\varepsilon^3), \qquad r \geq 1.$$

For the same two-position input $Q = K = (0, 1)^\top$, we have $C^r Q = (0, 2^{-r})^\top$. Forming the final comparison from $\widetilde{T}_{m-1}(\varepsilon Q)$ and $\widetilde{T}_{m-1}(\varepsilon K)$ therefore gives

$$\widetilde{\Lambda}_m(\varepsilon) = \frac{\varepsilon^2}{4^{m-1}} + O(\varepsilon^4),$$

and hence

$$\widetilde{\Lambda}_{m+1}(\varepsilon) - \widetilde{\Lambda}_m(\varepsilon) = -\frac{3}{4^m}\varepsilon^2 + O(\varepsilon^4).$$

Thus replacing the original branch values by the previous outputs can change the quadratic matching coefficient at every additional order. □

A finite-scale re-reading example. The local expansion above can be complemented by a finite-input example that isolates the values used by the second refinement. For a branch state U, write

$$\mathcal{B}(U) = \mathrm{softmax}\left(\frac{\rho(U)\rho(U)^\top}{\sqrt{d_h}} + M\right),$$

and define

$$T_1(U) = \mathcal{B}(U)U, \qquad T_2(U) = \mathcal{B}(T_1(U))U, \qquad \widetilde{T}_2(U) = \mathcal{B}(T_1(U))T_1(U).$$

Thus T_2 and $\widetilde{T}_2$ use exactly the same second-refinement weights; only their values differ.

Take $n = d_h = 2$, let $R_1 = I_2$ and

$$R_2 = \begin{pmatrix} \cos 1 & -\sin 1 \\ \sin 1 & \cos 1 \end{pmatrix}.$$

For $e = (1, 0)$ and $a > 0$, set

$$K_1 = ae, \qquad K_2 = \frac{a}{2} e R_2, \qquad Q = -K, \qquad \tau = \frac{a^2}{\sqrt{2}},$$

and let $P_U = \mathbf{1}U_1$ for $U \in \{Q, K\}$. The two visible logits in row 2 of the first self-refinement are $(\tau/2, \tau/4)$ for both branches. With $b = e^{-\tau/4}$, the attention weight on position 2 is at most b; hence

$$\|T_1(U) - P_U\|_{\infty,2} \le 2ab, \qquad U \in \{Q, K\}, \tag{32}$$

where $\|Z\|_{\infty,2} = \max_i \|Z_i\|_2$.

We next compare the second-refinement mixer with that of the constant-row reference. Every row of $T_1(U)$ is a convex combination of rows of U and therefore has norm at most a. Orthogonality of the positional rotations and Equation 32 give, for every visible score,

$$\left|\rho_i(T_1(U)_i)\rho_j(T_1(U)_j)^\top - \rho_i(U_1)\rho_j(U_1)^\top\right| \le 4a^2 b.$$

For finite softmax vectors, $\|\operatorname{softmax}(z) - \operatorname{softmax}(w)\|_1 \le \|z - w\|_\infty$; this follows from $(D \operatorname{softmax}(z)v)_k = p_k(v_k - \sum_s p_s v_s)$ and Cauchy–Schwarz. Since $a^2/\sqrt{2} = \tau$,

$$\|\mathcal{B}(T_1(U))_{i,:} - \mathcal{B}(P_U)_{i,:}\|_1 \le 4\tau b,$$

and consequently

$$\|T_2(U) - \mathcal{B}(P_U)U\|_{\infty,2} \le 4a\tau b. \tag{33}$$

By contrast, every row of $\widetilde{T}_2(U)$ is a convex combination of rows of $T_1(U)$, so

$$\|\widetilde{T}_2(U) - P_U\|_{\infty,2} \le 2ab. \tag{34}$$

For the constant-row reference, the row-2 self logits are $(\tau \cos 1, \tau)$. Hence the second mixer favors the current position by

$$\kappa = \tau(1 - \cos 1) > \frac{\tau}{4},$$

where $\cos 1 < 3/4$. Its weight on position 1 is at most $e^{-\kappa} \le b$, and therefore

$$\|\mathcal{B}(P_U)U - U\|_{\infty,2} \le 2ab.$$

Together with Equation 33,

$$\|T_2(U) - U\|_{\infty,2} \le a(4\tau + 2)b. \tag{35}$$

For visible candidates 2 and 1 at query position 2, define

$$\mathcal{L}(Y, Z) = \frac{\rho_2(Y_2)\big(\rho_2(Z_2) - \rho_1(Z_1)\big)^\top}{\sqrt{d_h}},$$

which equals their pairwise log-odds because the row-softmax normalizer cancels. Let

$$\Lambda^{(2)} = \mathcal{L}(T_1(Q), T_1(K)), \qquad \Lambda^{(3)} = \mathcal{L}(T_2(Q), T_2(K)), \qquad \widetilde{\Lambda}^{(3)} = \mathcal{L}(\widetilde{T}_2(Q), \widetilde{T}_2(K)).$$

The two reference preferences are

$$\mathcal{L}(Q, K) = \frac{\tau}{4}, \qquad \mathcal{L}(P_Q, P_K) = -\kappa.$$

If query-side states change row-wise by at most b_Q and key-side states by at most b_K, while their row norms are at most a, bilinearity bounds the change in $\mathcal{L}$ by $2a(b_Q + b_K)/\sqrt{2}$. Equations 32–35 therefore give

$$|\Lambda^{(2)} + \kappa| \le 8\tau b, \qquad |\widetilde{\Lambda}^{(3)} + \kappa| \le 8\tau b,$$

and

$$\left|\Lambda^{(3)} - \frac{\tau}{4}\right| \le 4\tau(4\tau+2)b.$$

For every finite $\tau \ge 64$,

$$(64\tau+32)e^{-\tau/4} < 1,$$

because the left side decreases on this range and is $4128e^{-16} < 1$ at $\tau = 64$. Hence

$$8\tau b < \frac{\tau}{4} < \kappa, \qquad 4\tau(4\tau+2)b < \frac{\tau}{4},$$

so

$$\mathcal{L}(Q,K) > 0, \qquad \Lambda^{(2)} < 0, \qquad \Lambda^{(3)} > 0, \qquad \widetilde{\Lambda}^{(3)} < 0. \tag{36}$$

These inequalities are strict at finite inputs, and therefore persist on an open neighborhood of each such (Q,K).

The construction isolates the role of the branch values. The first refinement makes the unrotated branch states nearly equal; after the destination-dependent rotations make the next mixer distinguish the two positions again, reading the original branch states can recover their distinction, whereas the same mixer reading the already-refined states remains near their common first-refinement state. This is a finite-scale mechanism complement to Proposition 3, not a monotonic benefit of increasing order.

Reusing earlier value states has precedents across Transformer layers: Value Residual Learning retains or shares early value representations, and NeuTRENO reintroduces initial-state information under repeated smoothing (Zhou et al., 2025; Nguyen et al., 2023). Here the original projected query/key states are reused specifically inside Nexus's recursive matching branches, and Equation 36 describes the resulting matching effect.

B Implementation and Evaluation Notes

Masking, positional encoding and heads. The softmax maps in the Nexus definition use the decoder's causal mask. RoPE is applied each time states form logits, and not to the mixed values (Su et al., 2021): base states use their source positions for inner logits, and refined states use their destination positions for final logits. The equations act per head. For GQA (Ainslie et al., 2023), each refined key head is shared among the query heads that shared it before conversion. The Qwen conversion follows this per-head convention.

Algorithm 1 Second-order Nexus for one head

Require: X, M, W_Q, W_K, W_V, positional map ρ
1: $Q \leftarrow XW_Q$, $K \leftarrow XW_K$, $V \leftarrow XW_V$
2: $B_Q \leftarrow \mathrm{softmax}(\rho(Q)\rho(Q)^\top/\sqrt{d_h} + M)$
3: $\widehat{Q} \leftarrow B_Q Q$
4: $B_K \leftarrow \mathrm{softmax}(\rho(K)\rho(K)^\top/\sqrt{d_h} + M)$
5: $\widehat{K} \leftarrow B_K K$
6: $A \leftarrow \mathrm{softmax}(\rho(\widehat{Q})\rho(\widehat{K})^\top/\sqrt{d_h} + M)$
7: **return** AV

B.1 Memoized Recursive Evaluation

For deterministic, dropout-free attention, define

$$T_0(U) = U, \qquad T_r(U) = \mathrm{Attn}_{M,\rho}(T_{r-1}(U), T_{r-1}(U), U), \quad r \ge 1. \tag{37}$$

The recursion in Equation 8 then becomes

$$H^m(U,Z,Y) = \mathrm{Attn}_{M,\rho}(T_{m-1}(U), T_{m-1}(Z), Y). \tag{38}$$

Indeed, $H^1(U,U,U) = T_1(U)$, and induction gives $H^r(U,U,U) = T_r(U)$. Each branch needs $m-1$ unique refinement cores, followed by one final comparison. For $m = 3$, the memoized graph has two query-side cores, two key-side cores and one final cross-core. Literal expansion makes seven calls because it evaluates identical child subexpressions separately.

Algorithm 2 Memoized order-m Nexus with deterministic, dropout-free attention

Require: X, W_Q, W_K, W_V, order $m \geq 1$, M, ρ
$Q_0 \leftarrow XW_Q$, $K_0 \leftarrow XW_K$, $V_0 \leftarrow XW_V$
$Q_{\text{cur}} \leftarrow Q_0$, $K_{\text{cur}} \leftarrow K_0$
for $r = 1, \ldots, m-1$ **do**
  $Q_{\text{cur}} \leftarrow \text{Attn}_{M,\rho}(Q_{\text{cur}}, Q_{\text{cur}}, Q_0)$
  $K_{\text{cur}} \leftarrow \text{Attn}_{M,\rho}(K_{\text{cur}}, K_{\text{cur}}, K_0)$
end for
return $\text{Attn}_{M,\rho}(Q_{\text{cur}}, K_{\text{cur}}, V_0)$

The original Q_0 and K_0 remain the values at every branch-refinement level, and V_0 is used only by the final comparison. At the 70M dimensions and $n = 2048$, the five-core schedule has analytical forward cost $1.88\times$ Standard. The experimental third-order result, 0.415, uses literal seven-core execution at $2.32\times$ Standard. Sharing subexpressions assumes deterministic, dropout-free calls; independently sampled attention-dropout masks make repeated calls different operations.

B.2 Incremental Second-Order Attention

Let q_t, k_t, v_t be the new unrotated projected states at position t. With attention dropout disabled, the two refined states are

$$\alpha^Q_{ta} = \frac{\exp(\rho_t(q_t)\rho_a(q_a)^\top/\sqrt{d_h})}{\sum_{u\leq t}\exp(\rho_t(q_t)\rho_u(q_u)^\top/\sqrt{d_h})}, \qquad \widehat{q}_t = \sum_{a\leq t}\alpha^Q_{ta}q_a, \tag{39}$$

$$\alpha^K_{tb} = \frac{\exp(\rho_t(k_t)\rho_b(k_b)^\top/\sqrt{d_h})}{\sum_{u\leq t}\exp(\rho_t(k_t)\rho_u(k_u)^\top/\sqrt{d_h})}, \qquad \widehat{k}_t = \sum_{b\leq t}\alpha^K_{tb}k_b. \tag{40}$$

The final comparison and value aggregation are

$$A_{tj} = \frac{\exp(\rho_t(\widehat{q}_t)\rho_j(\widehat{k}_j)^\top/\sqrt{d_h})}{\sum_{u\leq t}\exp(\rho_t(\widehat{q}_t)\rho_u(\widehat{k}_u)^\top/\sqrt{d_h})}, \qquad y_t = \sum_{j\leq t}A_{tj}v_j. \tag{41}$$

Append q_t, k_t, v_t to their raw histories before the inner operations, then append the newly formed $\widehat{k}_t$ before the final comparison. Each historical refined key depends only on its own causal prefix, so later tokens leave it unchanged. This second-order scheme uses the current refined query without retaining historical refined queries.

A direct cache stores raw Q, raw K, refined K, and V. With H_Q query heads and H_{KV} key/value heads of width d_h, its persistent elements per layer and historical token are

$$C_N = (H_Q + 3H_{KV})d_h, \qquad C_{\text{std}} = 2H_{KV}d_h, \qquad \frac{C_N}{C_{\text{std}}} = \frac{g+3}{2}, \quad g = H_Q/H_{KV}. \tag{42}$$

At equal precision, the listed state count is twice that of a standard KV cache for MHA ($g = 1$), and five times for the illustrative GQA grouping $g = 7$. These counts cover the listed persistent tensors, excluding temporary activations and separately stored rotated copies. Other storage choices can trade caching for recomputation. Equations 39–42 specify this direct implementation of the mathematical operator.

B.3 Experimental-Variant Definitions

The variants act at the attention sublayer, using its causal mask, head structure and output projection. Except for the hidden-state loop, the surrounding residual path is unchanged. Section 5.3 defines the branch-specific state-choice variants in Equations 11–14 and compares them with projection reuse. The hidden-state loop and Nexus–Diff are specified below.

For the hidden-state-loop control, let MHA_θ include the complete attention sublayer and its shared projections. Starting from $H^{(0)} = X$,

$$H^{(t+1)} = H^{(t)} + \text{MHA}_\theta(\text{LN}(H^{(t)})), \qquad t = 0, 1, 2. \tag{43}$$

Table 5: Task-level 70M design ablations. Macro-averages are computed before rounding task scores. Bold marks the highest value in each column.

Method	ARC-C	ARC-E	HSwag	LogiQA	PIQA	SciQ	Avg.
Standard	0.208	0.359	0.356	0.277	0.569	0.615	0.397
Nexus-Q	0.198	0.374	0.345	0.286	0.579	0.621	0.400
Nexus-QK	0.207	0.367	0.359	0.281	0.577	**0.663**	0.409
Nexus-QKV	**0.209**	0.380	0.354	**0.293**	0.579	0.636	0.409
Nexus (proj. reuse)	0.206	0.357	0.367	0.273	**0.601**	0.630	0.406
Nexus–Recursive	0.201	**0.391**	**0.368**	0.287	0.582	0.659	**0.415**
Hidden-state loop	0.200	0.373	0.346	0.283	0.583	0.625	0.402

Table 6: Public Pythia architecture dimensions used for the analytical FLOP accounting. All use training context 2048 and vocabulary size 50,304.

Scale	L	d	h	d_h
70M	6	512	8	64
160M	12	768	12	64
410M	24	1024	16	64
1B	16	2048	8	256

The feed-forward sublayer is applied once afterwards. The reported "2 loops" are two additional hidden-state updates after the base call, for three attention cores without additional projection parameters. Normalization, projections and residual updates are repeated along with these calls.

Nexus–Diff first obtains $\widehat{Q} = B_Q Q$ and $\widehat{K} = B_K K$, then uses Differential attention (Ye et al., 2025) for the final comparison. Split each refined head state into two equal channel groups, $(\widehat{Q}_1, \widehat{Q}_2)$ and $(\widehat{K}_1, \widehat{K}_2)$, with the corresponding $d_h/2$-dimensional rotary operators $\rho^{(s)}$. Then

$$A_s = \mathrm{softmax}\left(\frac{\rho^{(s)}(\widehat{Q}_s)\rho^{(s)}(\widehat{K}_s)^\top}{\sqrt{d_h/2}} + M\right), \quad s \in \{1, 2\}, \tag{44}$$

$$\text{Nexus–Diff}(X) = (A_1 - \lambda A_2)V, \tag{45}$$

using the learned λ parameterization and output normalization of Differential Transformer. This signed final map replaces the row-softmax map of default Nexus while retaining causal support. All experimental variants use the Standard run's attention-dropout setting; when dropout is enabled, different calls sample independent masks.

MTA control configuration. The reported six-layer MTA control is a scale-adapted instantiation rather than a verbatim reproduction of the 880M language-model configuration in Golovneva et al. (2025, Section 4.2). It applies query–key convolutions before and after softmax only in the third layer (zero-based index 2), with head mixing in all six layers. The published 880M setup applies query–key convolution every fourth layer and uses head groups of size $c_h = 16$; for the present eight-head control, the analytical accounting below uses all-head mixing ($c_h = h = 8$). Table 3 uses the corresponding single-layer count. Appendix B.4 also retains the rate-normalized $L/4$ reference in Equation 48.

B.4 Detailed Results and Analytical Costs

For sequence length n, hidden width d, depth L, feed-forward width $4d$ and vocabulary size V, dense-matrix accounting gives

$$\begin{aligned} F_{\mathrm{std}} &= L(24nd^2 + 4n^2 d) + 2ndV, \\ F(c) &= F_{\mathrm{std}} + 4(c-1)Ln^2 d, \\ F_{\mathrm{Nexus}} &= F_{\mathrm{std}} + 8Ln^2 d, \\ F_{\mathrm{Recursive}} &= F_{\mathrm{std}} + 24Ln^2 d. \end{aligned} \tag{46}$$

Table 7: Matched-scale analytical forward TFLOPs for one 512-token sequence, using the dense-matmul convention in Equation 46.

Scale	Standard	Nexus	Nexus/Standard
70M	.048922	.055365	1.1317
160M	.136197	.155525	1.1419
410M	.387755	.439295	1.1329
1B	.964489	1.033208	1.0712

Table 8: Cross-scale analytical dense-matmul forward costs for 160M Nexus and 410M Standard, derived from Equation 46. Values are TFLOPs per sequence; the final column is the fractional reduction relative to 410M Standard.

Context	160M Nexus	410M Standard	Ratio	FLOP reduction
512	.155525	.387755	.4011	59.89%
2048	.969992	1.860258	.5214	47.86%

A multiply–accumulate counts as two FLOPs. The language-model head is included; embedding lookups, normalization, nonlinearities and softmax are omitted. Here $F(c)$ describes projection-reuse configurations with c full-width attention cores per layer and one evaluation of the projections, feed-forward layers and language-model head. Default Nexus uses three cores and the third-order experiment uses seven. The dimensions in Table 6 give the matched-scale and cross-scale totals. Equation 46 uses equal-head MHA as in the Pythia models; GQA requires accounting for its query/key-head grouping.

Forward-equivalent training-compute proxy. For architecture a, let $F_a(n)$ be the analytical forward cost of one sequence under the corresponding accounting in this appendix. The Pythia experiments share $N_{\mathrm{upd}} = 143{,}000$ updates, $B_{\mathrm{seq}} = 1{,}024$ sequences per update and $n = 2048$. Define the cumulative forward-equivalent cost of one training run as

$$C_a^{\mathrm{FE}} := N_{\mathrm{upd}} B_{\mathrm{seq}} F_a(n).$$

The shared workload gives

$$\frac{C_a^{\mathrm{FE}}}{C_b^{\mathrm{FE}}} = \frac{F_a(n)}{F_b(n)}.$$

If κ is a common analytical multiplier from the counted forward work to forward-plus-backward work, define

$$C_a^{\mathrm{train,proxy}} := \kappa C_a^{\mathrm{FE}}.$$

Then

$$\frac{C_a^{\mathrm{train,proxy}}}{C_b^{\mathrm{train,proxy}}} = \frac{F_a(n)}{F_b(n)}.$$

The extension to training work is conditional on this common multiplier; architecture-dependent multipliers would change the ratio.

This proxy retains the full dense matrix dimensions and execution schedules used in the forward accounting. It excludes optimizer updates, activation recomputation, communication and the operations omitted from the forward count. It is distinct from measured runtime and cached autoregressive decoding. Accounting for all three training runs per condition multiplies the costs by the same factor and leaves their ratios unchanged.

Under this convention, the $1.44\times$ Nexus and $1.46\times$ 11-layer Standard entries provide a near-matched training-compute proxy comparison, whereas matched-scale Standard/Nexus training matches the token budget rather than compute. The forward-cost ratio of 160M Nexus to 410M Standard also applies to this proxy. Figure 2 remains a comparison of final-checkpoint endpoints, not a training trajectory.

The hidden-state loop repeats the four MHA projections as well as the attention core. Each of its two additional calls contributes $8nd^2 + 4n^2d$ per layer, giving

$$F_{\mathrm{loop}} = F_{\mathrm{std}} + 16Lnd^2 + 8Ln^2d. \tag{47}$$

The 11-layer Standard control uses F_{std} with $L = 11$ and the same width and vocabulary. These controls therefore have different cost expressions from the projection-reuse formula $F(c)$.

For the 70M dimensions, seven-core recursive Nexus costs 0.543582 TFLOPs, or $2.31959\times$ Standard, at $n = 2048$. At $n = 512$, the values are 0.068250 TFLOPs and $1.39506\times$. Kernel-launch overhead, memory traffic and decoding-state storage lie outside this forward arithmetic count.

For the reported MTA control, the analytical accounting uses $c_q \times c_k = 6 \times 11$ query–key convolutions and all-head mixing with $c_h = h = 8$. With N_{qk} query–key convolution layers and head mixing in all L layers, the analytical dense-matmul count is

$$F_{\mathrm{MTA}}^{\mathrm{layout}}(N_{\mathrm{qk}}) = F_{\mathrm{std}} + 4n^2h\left(N_{\mathrm{qk}}c_qc_k + Lc_h\right).$$

For $n = 2048$, $L = 6$, $d = 512$ and $V = 50{,}304$, Standard costs 0.234344 TFLOPs. The reported placement has $N_{\mathrm{qk}} = 1$ and costs 0.249645 TFLOPs, or $1.06529\times$ Standard, rounded to $1.07\times$ in Table 3. The rate-normalized reference replaces N_{qk} by $L/4$:

$$F_{\mathrm{MTA}} = F_{\mathrm{std}} + 4n^2h\left(\frac{L}{4}c_qc_k + Lc_h\right). \tag{48}$$

For the same dimensions, this gives 0.254074 TFLOPs and $1.08419\times$ Standard, rounded to $1.08\times$. For this control, both counts include the specified convolutions before and after softmax and exclude elementwise masking, normalization and gating.

Nexus–Diff has the same dense-matmul count as default Nexus: its two half-width final score products sum to one full-width product, and Equation 45 multiplies the combined map by V once. The scalar λ operation and output normalization are omitted with other elementwise operations.

B.5 Training and Evaluation Details

From-scratch training. The 70M, 160M, 410M and 1B models use the Pythia training budget and optimization settings (Biderman et al., 2023). Each run consists of 143,000 optimizer updates with a global batch of 1,024 sequences of length 2,048. This gives 2,097,152 sequence-token positions per update and a total budget of 299,892,736,000 positions, approximately 300B. Standard/Nexus pairs share the tokenizer and Pile data recipe.

For optimization, we follow the settings reported in Tables 1 and 6 of Biderman et al. (2023). We use Adam with $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\epsilon = 10^{-8}$, weight decay 0.01 and gradient clipping at 1.0. Peak learning rates are 10^{-3}, 6×10^{-4}, 3×10^{-4} and 3×10^{-4} for 70M, 160M, 410M and 1B, respectively. The schedule uses a warmup fraction of 0.01 (1,430 updates), cosine decay over the 143,000-update horizon, and a minimum learning rate of 10% of the peak. Attention and hidden-state dropout are both zero. The 70M design variants, attention controls, 11-layer Standard and hidden-state loop use the 70M training recipe and token budget; their reported results use the final training checkpoint.

Next-token evaluation. For the from-scratch models, next-token negative log-likelihood (NLL) is evaluated at the final training checkpoint. The evaluation data are sampled at random from the portion of the Pile outside these models' training set, using the same Pile version as in the Pythia study. The scored evaluation set contains 10B target tokens. Within each comparison, the three runs of each condition are scored on the same target tokens with identical conditioning contexts and target masks. The 70M controls follow the same matched-target evaluation protocol.

For a given comparison, let $\mathcal{I}$ index the scored target-token occurrences, with target x_t and evaluation context h_t for $t \in \mathcal{I}$, and let $N = |\mathcal{I}|$. For condition c and training run r, define

$$\mathcal{L}_{c,r} = -\frac{1}{N}\sum_{t\in\mathcal{I}} \ln p_{\theta_{c,r}}(x_t \mid h_t), \qquad \overline{\mathcal{L}}_c = \frac{1}{3}\sum_{r=1}^{3}\mathcal{L}_{c,r}.$$

Only scored targets enter the numerator and denominator, so each run-level loss is weighted by tokens rather than by batches or sequences. The reported perplexity is

$$\mathrm{PPL}_c = \exp\left(\overline{\mathcal{L}}_c\right),$$

computed from the unrounded mean NLL, not by averaging run-level perplexities. NLL is measured in nats per scored token; PPL is a re-expression of the same quantity.

Table 9: Final-checkpoint next-token evaluation. Within each scale, Standard and Nexus use the same scoring sample and context definition. NLL is averaged equally over three training runs; PPL in parentheses is derived from the unrounded mean NLL. The final column is the Nexus–Standard mean NLL difference.

Scale	Standard NLL (PPL)	Nexus NLL (PPL)	Δ NLL
70M	2.9026 (18.221)	2.8508 (17.302)	-0.0518
160M	2.5228 (12.463)	2.4534 (11.628)	-0.0694
410M	2.1903 (8.938)	2.1521 (8.603)	-0.0382
1B	2.0600 (7.846)	2.0265 (7.587)	-0.0335

Table 10: Final-checkpoint next-token evaluation for the 70M controls, using the same matched-target protocol and three-run NLL aggregation. PPL is derived from mean NLL. Relative analytical forward FLOPs are for $n = 2048$, normalized to six-layer Standard.

Method	Relative forward FLOPs	NLL (PPL)
Standard 6 layers	1.00×	2.9026 (18.221)
MTA (6L control)	1.07×	2.8683 (17.607)
Nexus (proj. reuse)	1.44×	2.8508 (17.302)
Standard 11 layers	1.46×	2.8668 (17.581)
Hidden-state loop	1.66×	2.8760 (17.743)
Nexus–Recursive	2.32×	2.8385 (17.090)

At 70M, Nexus has lower mean NLL than the 11-layer Standard and MTA controls on the shared scoring sample. The MTA control retains its lower analytical forward cost and the same displayed six-task average as Nexus in Table 3. Nexus–Recursive has the lowest mean NLL among these configurations and retains the literal seven-core execution cost of $2.32\times$ Standard.

Supervised fine-tuning. The 1.5B and 7B experiments start from Qwen2.5-Base and use the Open-R1 Qwen SFT configuration at revision `cec57f3`, replacing the Instruct checkpoint with the corresponding Base checkpoint. The SFT recipe uses the Bespoke-Stratos-17k dataset, response-only loss, one training epoch, a maximum sequence length of 4096 tokens, packed sequences, bf16 training, a learning rate of 2×10^{-5} with cosine scheduling and a warmup ratio of 0.1. Each pair starts from the same base checkpoint and uses matched data and SFT configurations. Evaluation uses the checkpoint from the final SFT update.

Run aggregation and decoding. Each pretraining, ablation, attention-control and Qwen SFT condition uses three independent end-to-end training runs. For condition c, run r and task b, let $s_{c,r,b}$ denote the separately evaluated checkpoint score. Let J_c be the number of tasks: six for the Pythia-style experiments and three for Qwen. The task means and run-level macro-averages are

$$\overline{s}_{c,b} = \frac{1}{3}\sum_{r=1}^{3} s_{c,r,b}, \qquad m_{c,r} = \frac{1}{J_c}\sum_{b=1}^{J_c} s_{c,r,b}.$$

The reported condition macro-average is

$$\overline{m}_c = \frac{1}{3}\sum_{r=1}^{3} m_{c,r} = \frac{1}{J_c}\sum_{b=1}^{J_c} \overline{s}_{c,b}.$$

These means are formed before display rounding. Each training run receives equal weight, as does each task in the macro-average. For Qwen, $s_{c,r,b}$ is the within-checkpoint score $a_{r,b}$ in Equation 49, evaluated separately for each condition; stochastic decodes are aggregated within that score rather than treated as additional training runs. Pythia-scale pairs use the same evaluation tasks; Qwen pairs use the same stochastic decoding settings.

Qwen decoding uses temperature 0.7, top-p 0.8 and top-k 20, with repetition penalties 1.10 and 1.05 for the 1.5B and 7B models. These settings follow the cited generation configurations.[1] For each training checkpoint, 16 independent decodes are drawn per problem. Let $e_{r,b,u,k} \in \{0, 1\}$ indicate answer correctness for training run r, task b, problem u and decode k: it is 1 for an answer scored as correct under task b's scoring rule and 0 otherwise. Let N_b be the number of problems in task b. The checkpoint and task-level averages are

$$a_{r,b} = \frac{1}{16N_b} \sum_{u=1}^{N_b} \sum_{k=1}^{16} e_{r,b,u,k}, \qquad \overline{a}_b = \frac{1}{3} \sum_{r=1}^{3} a_{r,b}. \tag{49}$$

Avg. is the unweighted mean of $\overline{a}_b$ across the three tasks. Thus the metric estimates single-generation correctness under the sampling protocol, rather than whether at least one of 16 responses is correct.

The setup fixes Open-R1 at revision `cec57f3`, MATH-500 to default/test at revision `2343f79`, and AIME24 to default/train at revision `58e32b4`; the cited entries give complete revision identifiers (Hugging Face, 2025; Hugging Face H4, 2024; 2025).

Cross-run ordering. Using the run-level macro-averages $m_{c,r}$ defined above, we compare each Nexus run with each baseline run within a setting. For Nexus condition N and baseline condition B, define

$$C_{\mathrm{N,B}} = \sum_{r=1}^{3} \sum_{s=1}^{3} \mathbf{1}[m_{\mathrm{N},r} > m_{\mathrm{B},s}] \,.$$

This summary uses all cross-run combinations rather than imposing a one-to-one pairing. Ties do not count as Nexus-higher orderings.

Table 11: Cross-run ordering of task macro-averages. $C_{\mathrm{N,B}}/9$ counts strict Nexus-over-baseline comparisons. Each macro-average uses six tasks for Pythia and three for Qwen. The nine comparisons reuse three trained checkpoints per condition; they are not nine independent training repetitions.

Setting	Baseline	Nexus-higher orderings
Pythia 70M	Standard 6 layers	9/9
Pythia 160M	Standard	9/9
Pythia 410M	Standard	9/9
Pythia 1B	Standard	9/9
Qwen 1.5B	Standard	9/9
Qwen 7B	Standard	9/9
Pythia 70M	Standard 11 layers	7/9
Pythia 70M	MTA (6L control)	5/9

For Qwen, the macro-averages use the within-checkpoint task scores in Equation 49; decoding samples are aggregated before the cross-run comparison. The counts describe the ordering of the observed run-level scores, not their dispersion or statistical significance. The MTA comparison retains the same displayed condition mean of 0.406 for both methods in Table 3.

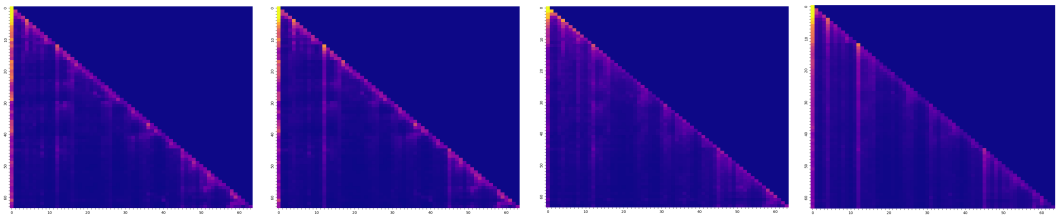
B.6 Attention-Map Illustration

Figure 3 places the two contextual refinement stages beside the final map used for value aggregation.

C Closest Operator-Level Comparisons

Table 12 compares where related mechanisms introduce context and how they form the map used for value aggregation. Nexus uses shared projections for separate causal query–query and key–key refinements, optionally recursive, followed by a final cross-branch comparison.

[1] Generation configurations: Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct.



(a) Standard (b) Nexus final (c) Inner Q map (d) Inner K map

Figure 3: Attention maps for the 64-token 70M example, averaged across checkpoints, layers and heads. Brightness is normalized separately within each panel. The panels show standard attention, the final Nexus map and its two inner refinement maps.

Table 12: Operator data paths. The third column describes the final matching or aggregation rule.

Operator	Context signal	Final matching or aggregation	Original setting
Standard	Token-local query–key score	Row-softmax of the base cross-score	General attention
Context-aware SAN (Yang et al., 2019)	Global, deep or lower-layer context injected into queries/keys	Query–key attention after contextual transformations	Encoder/decoder sequence modeling
MTA (Golovneva et al., 2025)	Causal convolution across query, key and head axes	Convolved query–key attention map	Causal language modeling
CASTLE (Song et al., 2026)	Earlier keys use the prefix through the current query	Row-softmax of combined causal and lookahead-key scores	Causal language modeling
JAT (Zhou et al., 2022)	Shared graph derived from the base score	Row-softmax of the two-sided propagated score	Causal/noncausal; multi-hop
SCLIP (Wang et al., 2024)	Separate query–query and key–key maps	Direct sum of the self-self maps	Training-free CLIP vision
GEM (Bousselham et al., 2024)	Iteratively refined query/key/value states	Ensemble of branch-specific self-self maps	Training-free visual localization
Nexus	Separate causal query–query and key–key maps	Final query–key attention over the two refined states	Causal decoder language modeling