Nexus: Higher-Order Attention Mechanisms in Transformers

Anonymous ACL submission

Abstract

Transformers rely on self-attention to model token dependencies, but the standard attention layer forms its logits from context-agnostic linear query and key projections. This design restricts each layer to pairwise interactions before the attention map is computed and can impose a fixed-subspace bottleneck on the attainable attention patterns. We propose **Nexus**, a higher-order attention mechanism that recursively refines query and key representations through causal inner attention before computing the final attention map. The resulting outer attention logits depend on contextualized query/key states and can incorporate multi-token interactions within a single layer. Nexus reuses the original projection matrices and introduces no additional projection parameters. We provide a theoretical analysis showing that standard linear query/key projections impose a fixed-subspace bottleneck, and that Nexus relaxes this constraint by making the effective query/key subspaces input-adaptive. Experiments on Pythia-scale language models and Qwen2.5 fine-tuning show consistent improvements across commonsense and reasoning benchmarks.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has become a foundational model for sequence modeling, achieving strong performance in natural language processing (Brown et al., 2020), computer vision (Dosovitskiy et al., 2020), scientific modeling (Jumper et al., 2021; Bi et al., 2023), and large-scale conversational systems (OpenAI, 2023). Its central component is self-attention, which enables each token to directly aggregate information from other tokens and thereby capture long-range dependencies.

Despite this success, standard self-attention has an important architectural restriction. In a conventional Transformer layer, the query and key vectors are obtained by token-wise linear projections, and the attention logit between positions i and j is computed as a pairwise inner product $Q_i K_j^\top$. Thus, before the attention map is formed, the score assigned to a token pair is not itself produced by a context-dependent reasoning process over other tokens. This can be limiting for tasks that require multi-hop reasoning, symbolic manipulation, or hierarchical dependency modeling.

A direct way to compensate for this limitation is to stack more Transformer layers, allowing later layers to reason over representations already mixed by earlier ones. However, increasing depth changes the whole block structure and typically adds both parameters and memory. Our goal is different: we aim to increase the reasoning density of a single attention layer by contextualizing the query/key states before the final attention map is formed, while keeping the projection-parameter count unchanged.

This perspective is particularly relevant for decoder-only language models. During autoregressive prediction, a token often needs to compare the current context with several earlier pieces of evidence, such as entities, variables, premises, or intermediate reasoning steps. If the query and key states are computed independently at each position before attention, the model must choose the final attention distribution before such evidence has been explicitly aggregated in the matching space. Nexus moves part of this aggregation before the final matching step: queries and keys become context-aware summaries, and the outer attention map is computed over these summaries rather than over purely local projections.

Several lines of work have attempted to improve Transformers. Efficient attention methods, such as Linformer (Wang et al., 2020), Performer (Choromanski et al., 2020), and Reformer (Kitaev et al., 2019), reduce the computational cost of self-attention through low-rank, kernelized, or hashing-

based approximations. These methods primarily target efficiency. Another line of work improves the expressive power of attention, for example through attention-on-attention (Huang et al., 2019), deformable attention (Xia et al., 2022), or high-order relation modeling (Wei et al., 2023; Hajij et al., 2022). These approaches demonstrate that richer interaction structures can be beneficial, although many such mechanisms have been more widely explored in vision, multimodal, or graph settings than in decoder-only language models.

In this work, we propose **Nexus**, a higher-order attention mechanism for language modeling and reasoning. Instead of using the linearly projected Q and K directly in the final attention computation, Nexus first refines them through causal inner self-attention. The refined query and key states are then used to form the outer attention map. Because the inner loops use the same projected Q and K states and introduce no new projection matrices, Nexus preserves the parameter count of the original attention layer while increasing the computational density of each layer.

2 Related Work

Efficient attention. The quadratic cost of self-attention has motivated many efficient Transformer variants. Linformer (Wang et al., 2020) approximates attention using low-rank projections, Performer (Choromanski et al., 2020) uses kernelized random feature approximations, and Reformer (Kitaev et al., 2019) uses locality-sensitive hashing. Sparse and local attention variants similarly reduce the number of token interactions. These methods are complementary to Nexus: they primarily reduce the cost of attention, whereas Nexus intentionally spends additional attention computation to increase the expressivity of query/key interactions.

Expressive and higher-order attention. Prior work has explored mechanisms that enrich attention beyond the standard pairwise formulation. Attention-on-Attention (Huang et al., 2019) composes attention operations, deformable attention (Xia et al., 2022) adapts the attended positions, and higher-order relation models (Wei et al., 2023; Hajij et al., 2022) explicitly encode interactions among multiple entities. Nexus follows the same broad motivation but targets decoder-only language models: it refines the query and key states through causal inner attention before the final attention map is computed.

Weight sharing and recurrent Transformers. Weight sharing across depth has been studied in models such as Universal Transformers (**?**), ALBERT (Lan et al., 2019), Perceiver-style iterative attention (Jaegle et al., 2021), block-recurrent Transformers (Hutchins et al., 2022), and Looped Transformers (Yang et al., 2024b). These models repeatedly apply shared modules to hidden states and can emulate iterative algorithms. Nexus differs in where recurrence is applied: rather than looping a full Transformer block over hidden states, it recursively refines the query/key spaces inside a single attention module. This distinction allows us to compare against a weight-shared recurrent Transformer baseline while isolating the benefit of query/key-specific refinement. Nexus is also compatible with architectural retrofitting, since existing query/key/value projections can be reused without adding new projection matrices.

3 Method

3.1 Preliminaries: Causal Self-Attention

Let $X \in \mathbb{R}^{n\times d}$ denote an input sequence of length n and hidden dimension d. For a single attention head with head dimension d_h, standard self-attention computes

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \quad (1)$$

where $W_Q, W_K \in \mathbb{R}^{d\times d_h}$ and $W_V \in \mathbb{R}^{d\times d_v}$. For decoder-only language modeling, we use a causal mask $M \in \mathbb{R}^{n\times n}$, where $M_{ij} = 0$ if $j \le i$ and $M_{ij} = -\infty$ otherwise. The row-wise causal attention operator is

$$\mathrm{Attn}_M(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top + M}{\sqrt{d_h}}\right) V. \quad (2)$$

For models using rotary position embeddings, we apply the RoPE operator $\rho(\cdot)$ to all query/key states used to form attention logits, while keeping the value states unrotated:

$$\mathrm{Attn}^\rho_M(Q, K, V) = \mathrm{softmax}\left(\frac{\rho(Q)\rho(K)^\top + M}{\sqrt{d_h}}\right) V. \quad (3)$$

All inner and outer attention operations in Nexus use the same causal mask, so the representation at position i depends only on positions $\le i$.

3.2 Motivation: Contextualized Query/Key Matching

In standard attention, the value aggregation is context-dependent, but the logits that decide the

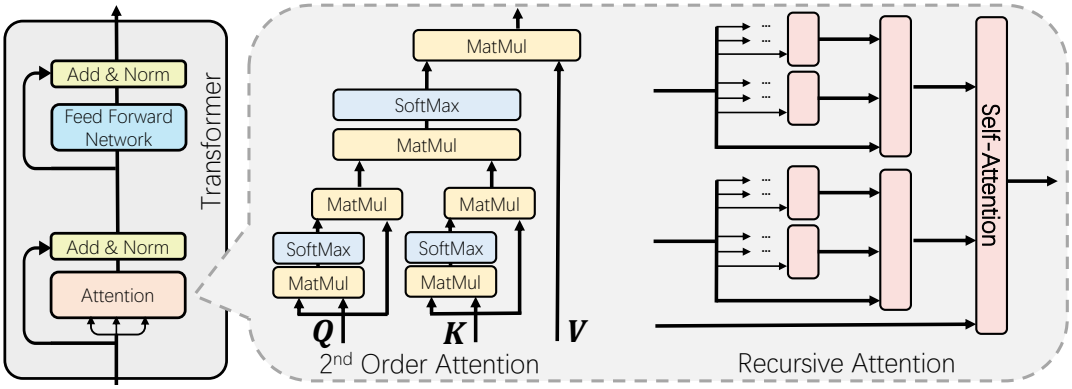


Figure 1: **Overview of Nexus. Left:** Nexus replaces the conventional self-attention module inside a Transformer block. **Middle:** In second-order Nexus, the query and key states are first refined by causal inner attention over themselves. The final outer attention is then computed from these contextualized query/key states. **Right:** The same idea extends to a tree-structured recursive formulation, where higher orders recursively refine the states used to form the final attention map.

aggregation are formed from token-wise query/key projections. As a result, a token must decide where to attend using a query vector that has not yet summarized the relevant context within the current layer. Nexus changes this order of operations. The model first lets the query and key states perform causal self-attention over their own spaces, producing context-aware matching vectors, and only then computes the final attention distribution. This preserves the usual value path while making the matching process itself more expressive.

3.3 Second-Order Nexus Attention

The default Nexus layer is a second-order attention mechanism. It first computes the standard projected states

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{4}$$

Instead of directly using Q and K to form the final attention map, Nexus refines them with causal inner attention:

$$\widehat{Q} = \mathrm{Attn}_M^{\rho}(Q, Q, Q), \tag{5}$$

$$\widehat{K} = \mathrm{Attn}_M^{\rho}(K, K, K). \tag{6}$$

The final output is then

$$\mathrm{Nexus}_2(X) = \mathrm{Attn}_M^{\rho}(\widehat{Q}, \widehat{K}, V). \tag{7}$$

This formulation introduces no additional projection matrices. The inner attention operations reuse the already projected Q and K states as their queries, keys, and values. Therefore, the parameter count of the attention layer remains the same as in standard self-attention.

Intuitively, the inner query attention allows the current position to gather evidence from previous positions before asking its final question, while the inner key attention lets source positions expose context-aware keys before being matched. This design keeps the value projection simple: the model still aggregates values once using the final outer attention, but the decision of which values to aggregate is made from refined query/key representations. This separation also makes the mechanism easy to retrofit into existing attention implementations, since the output interface of the attention layer remains unchanged.

Why this creates higher-order interactions. Let

$$\widehat{Q}_i = \sum_a \alpha_{ia}^{Q} Q_a, \quad \widehat{K}_j = \sum_b \alpha_{jb}^{K} K_b, \tag{8}$$

where α^Q and α^K are the causal inner attention weights. The outer attention logit between positions i and j becomes

$$\begin{aligned} S_{ij}^{\mathrm{Nexus}} &= \widehat{Q}_i \widehat{K}_j^{\top} \\ &= \sum_{a,b} \alpha_{ia}^{Q} \alpha_{jb}^{K} Q_a K_b^{\top}. \end{aligned} \tag{9}$$

In contrast to standard attention, where $S_{ij} = Q_i K_j^{\top}$ is a direct pairwise score, the Nexus score aggregates interactions over contextual token pairs (a, b). Thus, the final attention map can depend on other tokens before the outer attention distribution is formed.

Algorithm 1 Second-Order Nexus Attention for One Head

Require: Hidden states X, causal mask M, projections W_Q, W_K, W_V

$Q \leftarrow XW_Q$, $K \leftarrow XW_K$, $V \leftarrow XW_V$
$A_Q \leftarrow \mathrm{softmax}((\rho(Q)\rho(Q)^\top + M)/\sqrt{d_h})$
$\widehat{Q} \leftarrow A_Q Q$
$A_K \leftarrow \mathrm{softmax}((\rho(K)\rho(K)^\top + M)/\sqrt{d_h})$
$\widehat{K} \leftarrow A_K K$
$A \leftarrow \mathrm{softmax}((\rho(\widehat{Q})\rho(\widehat{K})^\top + M)/\sqrt{d_h})$
return AV

3.4 Tree-Structured Recursive Higher-Order Attention

We generalize Nexus through a tree-structured recursive formulation. Define the first-order operator as standard causal attention:

$$\mathcal{H}^1(Q, K, V) = \mathrm{Attn}_M^\rho(Q, K, V). \tag{10}$$

For $m \geq 2$, define

$$\mathcal{H}^m(Q, K, V) = \mathrm{Attn}_M^\rho(\mathcal{H}^{m-1}(Q, Q, Q), \mathcal{H}^{m-1}(K, K, K), V). \tag{11}$$

The m-th order Nexus layer is then

$$\mathrm{Nexus}_m(X) = \mathcal{H}^m(XW_Q, XW_K, XW_V). \tag{12}$$

The main experiments use $m = 2$, which corresponds to one recursive refinement of Q and K before the outer attention. We also evaluate $m = 3$ in the ablation study to test whether deeper recursive refinement further improves performance. This recursion is tree-structured: both the query branch and the key branch are expanded at each order. Thus, it is more expressive than a single linear loop, but its cost grows faster, which motivates our focus on the second-order case.

3.5 Parameter Sharing and Retrofitting

A naive implementation of higher-order attention could introduce separate inner projection matrices. We instead use a weight-shared design: all recursive computations are performed on the original projected Q, K, and V states, and no additional projection matrices are introduced. Consequently, Nexus has the same projection-parameter count as standard attention.

This design also makes Nexus suitable for retrofitting existing models. Given a pretrained Transformer layer, we retain its original W_Q, W_K, W_V and output projection parameters and replace the attention computation with the Nexus computation in Algorithm 1. Since the recursive refinement is parameter-free apart from the reused projections, the conversion does not introduce randomly initialized projection matrices or gates. The forward function changes because the query/key states are contextually refined, but the learnable attention parameters are inherited from the original model.

3.6 Complexity and Autoregressive Caching

Let $C = \mathcal{O}(n^2 d_h)$ denote the cost of one causal attention operation for one head. For the recursive definition above, the attention cost satisfies

$$T(1) = C, \quad T(m) = 2T(m-1) + C. \tag{13}$$

Therefore,

$$T(m) = (2^m - 1)C. \tag{14}$$

The default second-order Nexus layer has $T(2) = 3C$, corresponding to one inner attention for Q, one inner attention for K, and one outer attention. The third-order recursive variant has $T(3) = 7C$. In practice, we use $m = 2$ as the default because it provides a favorable trade-off between expressivity and computation.

The parameter complexity remains $\mathcal{O}(1)$ with respect to m because all recursive steps reuse the original projection matrices. However, Nexus increases training-time activation memory because the inner attention maps and refined query/key states must be stored for backpropagation. Gradient checkpointing can reduce this memory overhead.

For autoregressive decoding, causal masking ensures that the refined states of previous positions do not change when a new token is appended. Thus, Nexus supports KV caching. In addition to the final refined keys used by the outer attention, the implementation caches the base projected Q/K states needed to compute the inner refinement for the next token. This increases per-token decoding cost by a constant factor but preserves the standard left-to-right decoding semantics.

4 Theoretical Analysis

4.1 A Fixed-Subspace Bottleneck in Standard Attention

Prior work has studied low-rank bottlenecks in multi-head attention (Bhojanapalli et al., 2020).

Here we highlight a related but more specific limitation caused by token-wise linear query/key projections. The key observation is that softmax is invariant to adding a constant to every entry in a row of the logit matrix. Therefore, the relevant object is not only the rank of a logit matrix, but also the subspace in which each row of the log-attention pattern must lie. Standard linear projections force these rows to live in a fixed subspace determined by the input matrix X.

Proposition 4.1 (Fixed-subspace bottleneck)**.** *Let $X \in \mathbb{R}^{n\times d}$ and let standard attention use linear projections $Q = XW_Q$ and $K = XW_K$. If*

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right), \tag{15}$$

then for every row i, there exists a vector $u_i \in \mathbb{R}^d$ and a scalar $c_i \in \mathbb{R}$ such that

$$\log A_{i:} = Xu_i + c_i\mathbf{1}. \tag{16}$$

Equivalently, each row of $\log A$ lies in $\mathrm{Col}(X) + \mathrm{span}(\mathbf{1})$.

Corollary 4.2. *If $d < n-1$, then there exists a strictly positive row-stochastic matrix $A \in \mathbb{R}^{n\times n}$ with $\mathrm{rank}(\log A) = 1$ that cannot be represented by standard attention with linear query/key projections.*

The proof is provided in Appendix A. The result shows that the issue is not only the rank of the attention logits. Even a rank-one log-attention matrix may be unrepresentable when its rows fall outside the fixed subspace determined by the input matrix X and the all-one vector.

This formulation is useful because it separates two limitations that are sometimes conflated. Low rank constrains the dimensionality of the logit matrix, while the fixed-subspace constraint limits which directions each log-attention row can take. Nexus is designed to address the latter by making the effective query/key basis depend on the sequence itself.

4.2 How Nexus Relaxes the Bottleneck

In second-order Nexus, the refined key state can be written as

$$\widehat{K} = B_K(X)XW_K, \tag{17}$$

where

$$B_K(X) = \mathrm{softmax}\left(\frac{\rho(K)\rho(K)^\top + M}{\sqrt{d_h}}\right). \tag{18}$$

Similarly, $\widehat{Q} = B_Q(X)XW_Q$. Therefore, the final logits are computed from input-adaptive, causally mixed query/key states rather than from fixed token-wise linear projections. The standard constraint in Proposition 4.1 no longer applies in the same form: the effective subspaces are determined by $B_Q(X)XW_Q$ and $B_K(X)XW_K$, where $B_Q(X)$ and $B_K(X)$ are nonlinear functions of the sequence. This does not claim universal expressivity, but it formalizes why Nexus can mitigate the fixed-subspace bottleneck of standard attention. In particular, Nexus does not merely increase the rank of the final logit matrix; it changes the way the query/key subspaces are constructed by making them functions of the sequence itself. This is the main theoretical distinction from simply adding another static linear projection.

5 Experiments

We evaluate Nexus in two settings. First, we train Pythia-style decoder-only language models at multiple scales to test whether Nexus improves general language-model evaluation. Second, we retrofit Qwen2.5 models during supervised fine-tuning to evaluate whether Nexus can improve reasoning-oriented adaptation.

5.1 Evaluation on Different Scales

We use Pythia (Biderman et al., 2023) as the main pretraining-scale benchmark because its model suite, data, and training recipe are publicly documented. We follow the official Pythia training settings, including optimizer, learning-rate schedule, tokenizer, context length, and Pile (Gao et al., 2020) data setup. Evaluation is performed on six public benchmarks used in the Pythia evaluation suite: PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), SciQ (Welbl et al., 2017), ARC-E, ARC-C (Clark et al., 2018), and LogiQA (Liu et al., 2020). Unless otherwise stated, Nexus refers to the second-order shared QK variant, which introduces no additional projection parameters and uses three attention operations per layer.

Table 1 shows that Nexus improves average accuracy at all evaluated scales. The gains are broadly consistent across scale, suggesting that the benefit is not restricted to the smallest models. The average improvements are $+0.009$, $+0.015$, $+0.012$, and $+0.007$ for the 70M, 160M, 410M, and 1B models, respectively. At the 70M scale, the shared Nexus variant improves the average score from

Table 1: Main results: zero-shot accuracy comparison between Pythia baselines and second-order Nexus-QK-Shared across different model scales. The 70M Nexus row is aligned with the ablation result in Table 2. The best performance for each scale is highlighted in **bold**.

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

0.397 to 0.406 while using the same projection-parameter count as the baseline. The improvement is especially clear on PIQA and SciQ, while some retrieval-heavy or highly variable tasks such as ARC-C and LogiQA show smaller or negative changes at certain scales. Overall, the results support the view that query/key refinement provides a stable benefit, although it does not uniformly improve every dataset.

The benchmark-level pattern suggests that the gain is not tied to a single evaluation type. PIQA, SciQ, HellaSwag, and ARC-style tasks stress different combinations of physical commonsense, science knowledge, and sentence-level continuation. A single architectural change that improves the average across this mixture indicates that contextualized query/key matching can be useful as a general attention-layer modification. At the same time, the non-uniform per-task behavior motivates the ablations below, where we isolate which components of the attention computation should receive higher-order refinement.

5.2 Ablation Study

We conduct ablations at the 70M scale. Table 2 studies which projections should be refined, whether additional projection matrices are necessary, and how recursive depth affects performance. For readability, the table is split into two panels: Panel A summarizes configuration and efficiency, while Panel B reports the full benchmark scores. The non-shared rows are diagnostic variants that introduce separate inner projection matrices. Our default Nexus-QK-Shared uses no additional projection matrices.

Selection of higher-order components. Applying higher-order refinement only to the query states yields a modest improvement over the baseline. Refining both query and key states gives a stronger gain, indicating that the main benefit comes from improving the alignment space used to form attention logits. Refining values as well does not improve the average over Nexus-QK, suggesting that a simple value projection is sufficient and that query/key contextualization is the most important component.

Parameter efficiency via weight sharing. The shared QK variant removes all added projection matrices while keeping the same attention-operation count as the non-shared QK variant. Compared with Nexus-QK, Nexus-QK-Shared has the same $3.0\times$ attention-operation budget and lower projection FLOPs, while its average score drops only from 0.409 to 0.406. This small performance loss makes it the best parameter-efficient choice and the default setting for the main experiments.

Impact of recursive order. Increasing the recursive order from 2 to 3 improves the average score from 0.406 to 0.415, supporting the hypothesis that deeper query/key refinement can model richer dependencies. However, because the tree-structured cost grows as $2^m - 1$, the third-order variant requires $7.0\times$ attention operations. We therefore use second-order Nexus-QK-Shared as the default trade-off point.

Weight-shared recurrent Transformer baseline. To separate the effect of additional computation from the effect of query/key-specific refinement, we include a weight-shared recurrent Transformer baseline inspired by looped Transformer models (Yang et al., 2024b). This baseline repeatedly applies a shared Transformer attention layer to the hidden states under a matched $3.0\times$ attention-operation budget. It achieves an average score of 0.402, which is below Nexus-QK-Shared at 0.406,

Table 2: Ablation study of higher-order attention configurations on the 70M model. Panel A summarizes the configuration and efficiency of each variant. Panel B reports the full benchmark scores. **Proj.** denotes which states are recursively refined. **Shared** indicates whether the inner recursive attention reuses the original projections. **Added Proj.** counts additional projection sets relative to standard attention. **Attn Ops** is the attention-operation multiplier per layer. The best performance in each metric is highlighted in **bold**.

Panel A: Configuration and efficiency summary.

Model ID	Proj.	Shared	Order	Added Proj.	Attn Ops	Avg.
Baseline	–	–	–	0	1.0×	0.397
Effect of higher-order components						
Nexus-Q	Q	No	2	+1	2.0×	0.400
Nexus-QK	Q,K	No	2	+2	3.0×	0.409
Nexus-QKV	Q,K,V	No	2	+3	4.0×	0.409
Parameter efficiency and recursive depth						
Nexus-QK-Shared	Q,K	Yes	2	0	3.0×	0.406
Nexus-Recursive	Q,K	Yes	3	0	7.0×	**0.415**
Compute-matched recurrent baseline						
Looped TF (Yang et al., 2024b)	Hidden	Yes	2 loops	0	3.0×	0.402

Panel B: Full benchmark scores.

Model ID	ARC-C	ARC-E	HellaSwag	LogiQA	PIQA	SciQ	Avg.
Baseline	0.208	0.359	0.356	0.277	0.569	0.615	0.397
Effect of higher-order components							
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
Parameter efficiency and recursive depth							
Nexus-QK-Shared	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**
Compute-matched recurrent baseline							
Looped TF (Yang et al., 2024b)	0.200	0.373	0.346	0.283	0.583	0.625	0.402

although it performs better on a few individual datasets such as ARC-E and LogiQA. This result indicates that simply looping a shared attention layer is less effective on average than recursively refining the query/key spaces before the final attention map.

5.3 Visualization of Attention Patterns

To understand how Nexus differs from standard attention, we visualize average attention weights across layers and heads for the 70M models. We compare the baseline Pythia attention with the Nexus outer attention and the two inner attentions used to refine Q and K. The results are shown in Figure 2.

The baseline attention and the Nexus outer attention both preserve the causal language-model structure, including local diagonal patterns and attention to early tokens. This suggests that Nexus does not disrupt the basic sequential inductive bias of decoder-only Transformers.

The inner attention maps show different patterns from the outer attention. In particular, the inner key attention exhibits vertical structures, meaning that some positions are attended to by many later tokens during key refinement. We interpret this cautiously as evidence that the inner key attention may act as a context-dependent salience mechanism, highlighting positions that are useful for constructing the final key space. This visualization is qualitative and does not by itself prove semantic grounding; it is intended to illustrate that the inner refinement learns attention patterns distinct from the final outer attention.

5.4 Retrofitting Standard Transformers for Reasoning

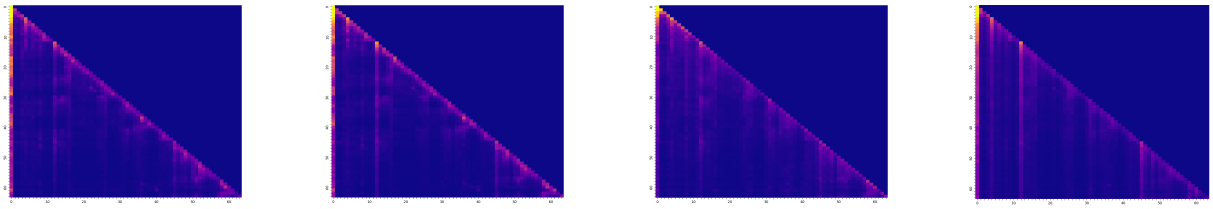
We further test whether Nexus can upgrade existing pre-trained Transformers during supervised fine-tuning. We use Qwen2.5-Base models (Yang et al., 2024a) at the 1.5B and 7B scales.

Setup. We use the Open-R1 framework[1] for supervised fine-tuning on reasoning-intensive data. The baseline directly fine-tunes the standard

[1] https://github.com/huggingface/open-r1

Table 3: Performance comparison of Qwen2.5 models fine-tuned with the standard architecture versus retrofitted Nexus architecture on reasoning benchmarks. **SFT** denotes standard supervised fine-tuning, while **Nexus-SFT** denotes fine-tuning after converting attention layers to Nexus-QK-Shared. Scores are mean accuracy over 16 stochastic sampling runs.

Base Model	Method	MATH-500	AIME24	GPQA-Diamond	Avg.
Qwen2.5-1.5B	Standard SFT	0.786	**0.194**	0.276	0.419
	Nexus-SFT	**0.801**	**0.194**	**0.280**	**0.425**
Qwen2.5-7B	Standard SFT	**0.921**	0.452	0.401	0.591
	Nexus-SFT	**0.921**	**0.475**	**0.407**	**0.601**



(a) Pythia Baseline (b) Nexus Outer (c) Nexus Inner (Q-Attn) (d) Nexus Inner (K-Attn)

Figure 2: Visualization of average attention heatmaps. The x-axis represents key positions, and the y-axis represents query positions. Brighter colors indicate higher attention weights. (a) Standard self-attention in Pythia. (b) The outer main attention of Nexus. (c) The inner recursive attention used to refine queries. (d) The inner recursive attention used to refine keys.

Qwen2.5 model. For Nexus-SFT, we convert each attention layer to second-order Nexus-QK-Shared by reusing the original W_Q, W_K, W_V and output projection parameters. No additional projection matrices or gates are introduced. We evaluate on MATH-500, AIME24, and GPQA-Diamond.

For evaluation, we use stochastic decoding and run 16 independent samples per problem. We follow the released Qwen2.5 generation configuration, using sampling rather than greedy decoding. Scores in Table 3 are reported as mean accuracy over the 16 sampling runs under the same decoding settings for the baseline and Nexus-SFT.

Results. Table 3 shows that Nexus-SFT gives consistent but modest improvements. At the 1.5B scale, Nexus improves MATH-500 and GPQA-Diamond while matching the baseline on AIME24. At the 7B scale, the largest improvement appears on AIME24, suggesting that query/key refinement can help competition-style mathematical reasoning under the same fine-tuning setup. Because the conversion reuses the original attention projections, these gains do not come from increasing the number of trainable projection parameters. The results support the view that Nexus can serve as a parameter-preserving architectural retrofit for existing models, although the gains should be interpreted together with the increased attention computation.

The use of 16 stochastic samples is intended to reduce sensitivity to a single decoding trajectory, which is especially important for small reasoning benchmarks such as AIME24. Since both the baseline and Nexus-SFT use the same sampling configuration, the comparison isolates the effect of the attention architecture under the same fine-tuning and decoding protocol.

6 Conclusion

We introduced Nexus, a higher-order attention mechanism that recursively refines query and key states before computing the final attention map. Unlike standard self-attention, where query and key vectors are token-wise linear projections, Nexus makes the effective query/key states context-dependent through causal inner attention. This allows the final attention logits to incorporate multi-token interactions within a single layer. The default Nexus-QK-Shared variant introduces no additional projection parameters and uses a $3\times$ attention-operation budget. Theoretical analysis shows that standard linear query/key projections impose a fixed-subspace bottleneck, while Nexus relaxes this constraint through input-adaptive query/key subspaces. Empirical results on Pythia-scale models and Qwen2.5 reasoning fine-tuning show consistent improvements, making Nexus a parameter-efficient but compute-heavier alternative to standard attention.

Limitations

Nexus increases attention FLOPs and activation memory. Although the shared design keeps the projection-parameter count unchanged, it does not remove the additional inner attention operations required to refine query and key states. The default second-order variant uses approximately three attention operations per layer, and the tree-structured recursion grows as $2^m - 1$. This may limit deployment in latency-sensitive, resource-constrained, or very long-context settings unless optimized kernels, sparse inner attention, or checkpointing strategies are used. Another practical direction is to apply Nexus only to selected layers, such as middle or upper layers where contextual matching may be most useful. In addition, our attention visualizations are qualitative; future work should include quantitative analyses such as attention entropy, long-range attention mass, and token-level salience studies.

References

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Low-rank bottleneck in multi-head attention models. In *International conference on machine learning*, pages 864–873. PMLR.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2023. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, and 1 others. 2020. Rethinking attention with performers. In *International Conference on Learning Representations*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, and Karthikeyan Natesan Ramamurthy. 2022. Higher-order attention networks. *arXiv preprint arXiv:2206.00606*, 2(3):4.

Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4634–4643.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *Advances in neural information processing systems*, 35:33248–33261.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, and 1 others. 2021. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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

OpenAI. 2023. Chatgpt (mar 14 version). https://chat.openai.com/chat.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Xi Wei, Zhangxiang Shi, Tianzhu Zhang, Xiaoyuan Yu, and Lei Xiao. 2023. Multimodal high-order relation transformer for scene boundary detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22081–22090.

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

Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. 2022. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4794–4803.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. 2024b. Looped transformers are better at learning learning algorithms. In *International conference on learning representations*, volume 2024, pages 42195–42214.

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

A Proofs

Proof of Proposition 4.1. For standard attention, the unnormalized logit row for position i is

$$\frac{Q_i K^\top}{\sqrt{d_h}} = \frac{Q_i W_K^\top X^\top}{\sqrt{d_h}}. \tag{19}$$

After transposing the row vector, this logit vector lies in $\mathrm{Col}(X)$. More explicitly, there exists $u_i \in \mathbb{R}^d$ such that

$$\left(\frac{Q_i K^\top}{\sqrt{d_h}}\right)^\top = X u_i. \tag{20}$$

Let $z_i^\top = Q_i K^\top / \sqrt{d_h}$. Since $A_{ij} = \exp(z_{ij}) / \sum_\ell \exp(z_{i\ell})$, we have

$$\log A_{i:} = z_i - \log\left(\sum_\ell \exp(z_{i\ell})\right) \mathbf{1}^\top. \tag{21}$$

Therefore each row of $\log A$ belongs to $\mathrm{Col}(X) + \mathrm{span}(\mathbf{1})$. □

Proof of Corollary 4.2. If $d < n - 1$, then the subspace $\mathrm{Col}(X) + \mathrm{span}(\mathbf{1})$ has dimension at most $d + 1 < n$. Hence, there exists a vector $a \in \mathbb{R}^n$ such that

$$a \notin \mathrm{Col}(X) + \mathrm{span}(\mathbf{1}). \tag{22}$$

Define a strictly positive probability vector

$$p_j = \frac{\exp(a_j)}{\sum_{\ell=1}^n \exp(a_\ell)}. \tag{23}$$

Let $A = \mathbf{1} p^\top$. Then A is strictly positive and row-stochastic. Moreover,

$$\log A = \mathbf{1} (\log p)^\top, \tag{24}$$

which has rank one as long as $\log p$ is nonzero. Suppose, for contradiction, that A can be represented by standard attention with linear query/key projections. By Proposition 4.1, each row of $\log A$ must lie in $\mathrm{Col}(X) + \mathrm{span}(\mathbf{1})$. But

$$\log p = a - \log\left(\sum_{\ell=1}^n \exp(a_\ell)\right) \mathbf{1}, \tag{25}$$

which is not in $\mathrm{Col}(X) + \mathrm{span}(\mathbf{1})$ because this subspace already contains $\mathbf{1}$ and a was chosen outside it. This contradiction completes the proof. □

B Additional Implementation Details

Causal masking. All Nexus attention operations use the same causal mask as the original decoder-only Transformer layer. This includes the inner query refinement, the inner key refinement, and the final outer attention. Consequently, the refined query and key at position i only depend on positions at or before i. This is important because the inner attention loops otherwise would create an unintended information path from future tokens to earlier positions during language-model training.

Rotary position embeddings. For models using RoPE, the rotary transformation is applied whenever query/key states are used to form attention logits. In second-order Nexus, this means that RoPE is applied to the query/key states in the inner Q-attention, the inner K-attention, and the final outer attention. Value states are not rotated. This mirrors the convention used in standard decoder-only attention while extending it consistently to the additional inner attention operations.

Autoregressive caching. Because all attention operations are causal, the refined states for previous positions do not change when a new token is appended. During decoding, an implementation can therefore cache the base projected query/key states required by the inner refinement, as well as the refined key states used by the outer attention. The per-token decoding cost increases by a constant factor relative to standard attention, but the left-to-right semantics and compatibility with KV caching are preserved.

C Additional Notes on the Recurrent Baseline

The compute-matched recurrent baseline in Table 2 of the main paper is designed to test whether the gains of Nexus are merely due to spending more attention computation. Inspired by looped Transformer models (Yang et al., 2024b), this baseline repeatedly applies a shared attention computation to hidden states under a matched $3.0\times$ attention-operation budget. It therefore has the same projection-parameter count as the original model and the same attention-operation multiplier as second-order Nexus-QK-Shared.

The key distinction is where the repeated computation occurs. The recurrent baseline loops over hidden states, while Nexus performs the additional

computation specifically inside the query/key construction pathway before the final attention map is formed. This makes the comparison a targeted test of the proposed mechanism: if additional computation alone were sufficient, the hidden-state recurrent baseline should match the Nexus result. In our ablation, the recurrent baseline improves over the standard baseline but remains below Nexus-QK-Shared on average, suggesting that query/key-specific refinement is more effective than a generic hidden-state loop at the same attention-operation budget.

D Evaluation Protocol Notes

For Pythia-scale experiments, we follow the official Pythia training and evaluation setup as closely as possible, including the tokenizer, context length, optimizer schedule, and Pile data recipe. The purpose of this setup is to isolate the architectural change while avoiding confounds from a different training corpus or evaluation suite.

For Qwen2.5 supervised fine-tuning experiments, both the standard SFT model and the Nexus-SFT model use the same data, fine-tuning recipe, answer extraction procedure, and generation configuration. We evaluate each problem with 16 stochastic samples and report mean accuracy. This sampling protocol reduces sensitivity to a single decoding path, which is useful for small and high-variance reasoning benchmarks such as AIME24. The comparison is therefore between two architectures under the same fine-tuning and decoding conditions, rather than between different prompting or sampling strategies.