KRONECKER PRODUCT ATTENTION: MEMORY-EFFICIENT KV COMPRESSION VIA STRUCTURED FACTORIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient autoregressive generation is increasingly constrained by the memory footprint of the key–value (KV) cache, which grows linearly with sequence length, batch size, and model depth. To reduce this memory footprint, existing attention architectures use head sharing in Multi-Query Attention (MQA) and Grouped-Query Attention (GQA), latent compression in Multi-Head Latent Attention (MLA), and token-dependent factorization in Tensor Product Attention (TPA). We propose Kronecker Product Attention (KPA), which represents query, key, and value activations as sums of Kronecker products of compact, token-dependent matrix factors. By coupling head and feature dimensions within each factor, KPA allows different heads to receive distinct feature directions within a single Kronecker term. KPA recovers standard TPA as a special case and supports exact attention computation directly from the compact factors. A Kronecker-separable rotary operator preserves the relative-position identity while rotating only one compact factor of each query and key term, without an additional rotary-key cache branch. Across four model scales, from 125M to 1.5B parameters, KPA achieves higher average zero-shot accuracy than MHA, MQA, GQA, MLA, and TPA on nine benchmarks, while caching 6–10× fewer KV entries than MHA with the same head shape. The Kronecker head split provides a design parameter for trading accuracy against cache size.

1 INTRODUCTION

The key–value (KV) cache can become a substantial memory cost in autoregressive language-model inference, particularly with long contexts or large serving batches (Kwon et al., 2023; Liu et al., 2024). It retains representations of earlier tokens' keys and values for reuse during decoding. For a fixed representation size and storage precision, the cache for each sequence and attention layer grows linearly with the number of retained tokens. These storage requirements accumulate across layers and concurrent requests. Where KV storage limits serving capacity, reducing the state stored per token can make room for longer contexts or more concurrent requests, However, obtaining these savings while maintaining predictive accuracy is challenging.

Architectural approaches to KV-cache compression make different choices about how keys and values are represented. Multi-query attention (MQA) (Shazeer, 2019) uses one key head and one value head shared across all query heads. Grouped-query attention (GQA) (Ainslie et al., 2023) partitions query heads into groups, each sharing its own key head and value head. Multi-head latent attention (MLA), as introduced in DeepSeek-V2 (DeepSeek-AI, 2024), caches a compressed key–value latent vector together with a separate rotary-key component. Tensor product attention (TPA) (Zhang et al., 2025) constructs query, key, and value matrices from token-dependent factors. In standard TPA, each term is the outer product of a vector over heads and a vector over within-head features. Its higher-order extension further factorizes the feature vector while keeping the full head index in a separate vector (Zhang et al., 2025, Appendix C). These designs motivate a more specific question: *how should head and feature indices be grouped within compact, token-dependent factors?*

We propose **Kronecker Product Attention (KPA)**, which constructs query, key, and value matrices as sums of Kronecker products of compact matrix factors computed from each token's hidden state.

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.

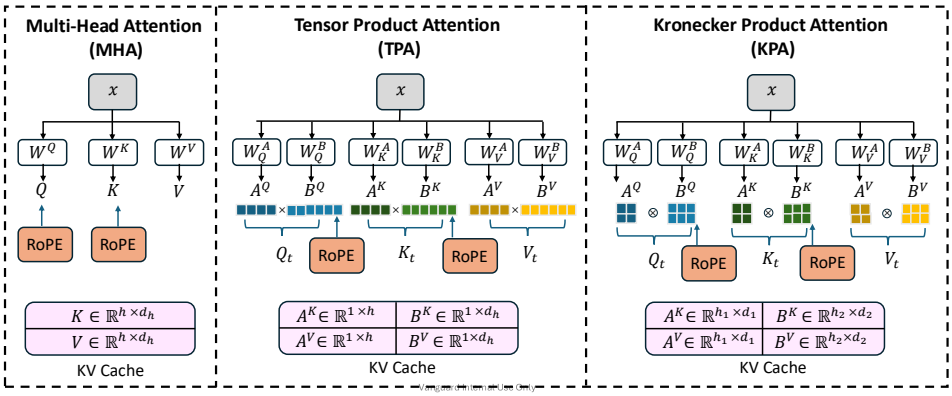

Figure 1: Overview of multi-head attention (MHA), tensor product attention (TPA), and Kronecker Product Attention (KPA). The factorized panels illustrate one representative term; token and rank indices, summation, and normalization are omitted. MHA projects the token state directly into dense query, key, and value representations. Standard TPA combines contextual head and feature vectors through an outer product (shown as $\times$), whereas each KPA matrix factor combines a head sub-index with a feature sub-index and the factors form a Kronecker product ($\otimes$). In the KPA panel, RoPE denotes the Kronecker-separable operator defined in Theorem 3.4, equivalently implemented by rotating only the query and key B factors. Key B factors shown in the cache are understood to include this rotation. Values remain unrotated, and queries are not cached. Cache boxes show factor shapes per rank term.

Each factor is indexed by both a head sub-index and a feature sub-index, so the Kronecker product of two small factors yields a full-size, block-structured matrix with the same shape as a standard key or value matrix. This design is motivated by analyses showing that heads within a layer often form groups with similar behavior (Clark et al., 2019): KPA shares factors within groups of heads while still letting individual heads differ. Only the key and value factors are cached and the full key and value matrices never need to be stored. Figure 1 illustrates this representation and cache layout. Kronecker structure has previously been used to compress weight matrices (Tahaei et al., 2022; Edalati et al., 2022), to parameterize fine-tuning updates (Edalati et al., 2025), and to build multiway attention for encoder models over tensor-structured inputs (Omranpour et al., 2024), where it reduces training-time computation; KPA instead applies it to the token-dependent key and value activations stored in the cache during autoregressive decoding.

The difference from standard TPA lies in how heads share feature vectors. In TPA, every head's key is built from the same few feature vectors, scaled by head-specific coefficients, so all heads' keys at a token lie in a subspace whose dimension is at most the number of terms. In KPA, each head receives the Kronecker product of one row from each factor, so different heads can receive different feature directions even within a single term. This gives KPA more diversity across heads at the same number of cached terms, and standard TPA is recovered as the special case in which the factors reduce to vectors. The factor shapes also set the storage cost: the cache grows with the sum of the factor sizes rather than their product, and it is smallest when the two factors are balanced. The split therefore lets practitioners trade head diversity against cache size.

The Kronecker product offers three useful properties at once. First, it turns two small factors into a full-size matrix whose block structure matches how attention splits into heads and features. Second, its mixed-product identity breaks each query–key inner product into a product of two smaller inner products. Third, it supports a rotary operator that acts on only one factor. Using these properties, we compute attention scores and outputs directly from the factors; the result is exactly the same as applying causal softmax attention to the reconstructed matrices. For rotary position embeddings (RoPE) (Su et al., 2024), we extend the construction used in higher-order TPA (Zhang et al., 2025, Appendix C.1) to matrix factors. Our operator rotates only one factor of each query and key term and reuses the same frequencies in every feature block. It still preserves the key property of RoPE: the rotation in each query–key inner product depends only on how far apart the two tokens are. Keys

can therefore be cached already rotated, without changing the factor shapes or adding the separate rotary-key component that MLA requires.

We train 11 KPA configurations in total across four nominal model scales, from 125M to 1.5B parameters, and evaluate them on nine zero-shot benchmarks. We compare their task scores with those reported by Zhang et al. (2025) for standard multi-head attention (MHA), MQA, GQA, MLA, and standard TPA. At each scale, the best evaluated KPA configuration has the highest unweighted nine-task average among these methods, with gains of 0.83 to 1.99 percentage points over TPA. As can be seen, gains are not uniform across individual tasks.

For the evaluated factor shapes, analytical cache counts give compression ratios of 6–10$\times$ relative to MHA with the same head count and head width, assuming factor-only KV caching. These ratios compare stored scalar entries per token and layer, not measured device memory. At 773M, the more balanced head split uses 25% fewer KV entries than the highest-average KPA configuration at that scale, with an average accuracy 0.58 percentage points lower (Figure 2). The head split thus exposes distinct accuracy–storage operating points within KPA.

Our contributions are:

- **A matrix-factor attention architecture.** KPA groups head and feature indices within token-dependent matrix factors, with standard TPA recovered as a special case.
- **Exact factored attention and a compatible rotary operator.** We derive score and output computations from the factors and specify a Kronecker-separable rotary operator that preserves the relative-position identity without an additional rotary-key cache component.
- **An evaluation of accuracy and analytical KV storage.** We compare eleven configurations across four model scales with five published accuracy baselines and examine the accuracy and storage differences between head splits.

2 Related Work

Architectural KV compression. Several attention architectures reduce KV-cache storage by changing the representation retained for each token. Multi-query attention (MQA) (Shazeer, 2019) uses one key head and one value head shared across all query heads, while grouped-query attention (GQA) (Ainslie et al., 2023) shares them within groups of query heads. Multi-head latent attention (MLA) (DeepSeek-AI, 2024) instead caches a compressed key–value latent representation together with a separate rotary-key component. More recent architectures explore other forms of compression and reuse. MFA and MFA-KR (Hu et al., 2025) factorize parts of the attention parameterization, with MFA-KR reusing the key cache in the value computation, while GTA and GLA (Zadouri et al., 2025) explore key–value reuse and grouped latent representations for efficient decoding. These methods modify head sharing, latent representations, or reuse between keys and values. KPA instead changes how head and feature indices are grouped within token-dependent factors. MFA/MFA-KR and GTA/GLA are related architectural approaches, but are not experimental baselines in this work.

Tensor Product Attention. TPA (Zhang et al., 2025) is the closest prior work to KPA. It constructs query, key, and value matrices from compact, token-dependent factors and retains key and value factors in the KV cache. In standard TPA, each term is the outer product of a vector indexed by heads and a vector indexed by within-head features. Higher-order TPA further factorizes the feature vector while retaining the complete head index in a separate vector (Zhang et al., 2025, Appendix C). KPA uses a different index grouping: each of its two matrix factors contains one head sub-index and one feature sub-index. Consequently, within a single KPA term, different heads can receive different feature directions, whereas a single standard TPA term provides head-specific scalings of one shared feature vector. This comparison concerns individual terms; after summing multiple terms, standard TPA can also produce different vectors across heads. Standard TPA is recovered by the KPA factorization when one factor reduces to a head vector and the other to a feature vector (Section 3.1). Table 1 summarizes the index groupings of standard TPA, higher-order TPA, and KPA.

Kronecker and structured parameterizations. Kronecker structure has been used in transformers for purposes distinct from KV-cache compression. KroneckerBERT and KnGPT2 (Tahaei et al., 2022; Edalati et al., 2022) factorize learned weight matrices, while KronA (Edalati et al.,

2025) uses Kronecker-structured updates for parameter-efficient adaptation. Tensorized neural networks (Novikov et al., 2015) and Monarch (Dao et al., 2022) likewise impose structured representations on model parameters. These methods structure static weights or weight updates rather than the token-dependent key and value representations stored during decoding. Higher-Order Transformers (HOT) (Omranpour et al., 2024) use Kronecker structure inside the attention operator for tensor-structured inputs, combining attention along different input axes. KPA instead applies Kronecker factorization to the head-by-feature representation associated with each token. Thus, although the same algebraic product appears, the object being factorized and the source of the efficiency are different.

Token retention, quantization, and cache management. A separate line of work reduces KV memory without changing the attention representation in the same way. H2O (Zhang et al., 2023) retains heavy-hitter and recent tokens, while StreamingLLM (Xiao et al., 2024) keeps initial attention-sink tokens together with a recent window. These approaches reduce the number of tokens retained in the cache. KIVI (Liu et al., 2024) and KVQuant (Hooper et al., 2024) instead reduce the precision of cached states. PagedAttention (Kwon et al., 2023) improves KV-cache memory management through block-based allocation and sharing, without changing the representation stored for each token. KPA targets a different part of the storage cost: the number of scalar entries retained per token and layer under factor-only caching. Retention, quantization, and allocation techniques are therefore potentially complementary to KPA, although their combination with KPA is not evaluated here.

Rotary encoding for factorized representations. RoPE (Su et al., 2024) applies position-dependent rotations to query and key features and yields a relative-position identity for their inner products. Standard TPA applies RoPE to its feature factors, and higher-order TPA derives a separable rotary construction in which the rotation acts on one constituent feature factor (Zhang et al., 2025, Appendix C.1). KPA adapts this idea to its matrix factors. The operator defined in Section 3.3 rotates the feature dimension of the second query and key factor and preserves the relative-position identity while leaving the factor shapes unchanged. Rotated key factors can therefore be retained directly in the cache without an additional rotary-key component.

3 Kronecker Product Attention

Standard multi-head attention caches dense key and value matrices for every token, with no constraint relating one head's entries to another's. KPA instead builds each query, key, and value matrix from two small matrices computed from the token's hidden state, each spanning part of the head index and part of the feature index. Their Kronecker product tiles scaled copies of one factor according to the entries of the other, producing a full-size matrix with block structure, and summing several such products lets these block patterns combine. Heads therefore share factors while still receiving distinct key and value vectors, unlike the head sharing in MQA and GQA. Only the key and value factors are cached, so storage grows with the size of the factors rather than with the full attention width.

Section 3.1 formalises the factorisation and its relation to TPA. Section 3.2 derives exact score and output contractions from the factors. Section 3.3 gives a rotary operator that acts only on the compact query and key factors. Section 3.4 counts the resulting cache, parameters, and computation.

3.1 Notation and Factorisation

Notation. Consider an attention layer with h heads of width d_h and model width d_{model}. Bold uppercase letters denote matrices, bold lowercase letters denote column vectors, and unbolded letters denote scalars and indices. Subscripts t and s index token positions and r and u index Kronecker terms. The superscript $Z \in \{Q, K, V\}$ identifies the query, key, or value representation. We write $[n] = \{1, \dots, n\}$, $\boldsymbol{I}_n$ for the $n \times n$ identity, and $\boldsymbol{M}[p, :]$ for row p of $\boldsymbol{M}$.

Definition 3.1 (Kronecker product)**.** *For $\boldsymbol{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\boldsymbol{B} \in \mathbb{R}^{m_2 \times n_2}$, the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$ is*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} A_{11}\boldsymbol{B} & \cdots & A_{1n_1}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ A_{m_1 1}\boldsymbol{B} & \cdots & A_{m_1 n_1}\boldsymbol{B} \end{pmatrix}.$$

Representation	Entry of one term	Factor shapes
Ordinary TPA	$a_{(i_1,i_2)}\, b_{(j_1,j_2)}$	$h;\ d_h$
Third-order TPA	$a_{(i_1,i_2)}\, c_{j_1}\, b_{j_2}$	$h;\ d_1;\ d_2$
KPA	$A_{i_1,j_1}\, B_{i_2,j_2}$	$h_1 \times d_1;\ h_2 \times d_2$

Table 1: Index grouping in one term of each contextual factorisation, with $h = h_1 h_2$ and $d_h = d_1 d_2$. All factors depend on the token state; token, representation, and rank indices are omitted. TPA keeps all head indices in a single vector, whereas each KPA factor pairs a head sub-index with a feature sub-index.

For conformable matrices, it satisfies the mixed-product identity (Van Loan & Pitsianis, 1993)

$$(\boldsymbol{M} \otimes \boldsymbol{N})(\boldsymbol{P} \otimes \boldsymbol{S}) = (\boldsymbol{M}\boldsymbol{P}) \otimes (\boldsymbol{N}\boldsymbol{S}), \tag{1}$$

which underlies the factored score computation (Section 3.2) and the rotary construction (Section 3.3).

Factorisation. Let $h = h_1 h_2$ and $d_h = d_1 d_2$. We index heads and features by $i = (i_1 - 1)h_2 + i_2$ and $j = (j_1 - 1)d_2 + j_2$, with $i_1 \in [h_1]$, $i_2 \in [h_2]$, $j_1 \in [d_1]$, and $j_2 \in [d_2]$, so that $(\boldsymbol{A} \otimes \boldsymbol{B})_{i,j} = A_{i_1,j_1} B_{i_2,j_2}$. For token state $\boldsymbol{x}_t \in \mathbb{R}^{d_{\text{model}}}$ and $r \in [R_Z]$, the factors are

$$\boldsymbol{A}^Z_{t,r} = \text{reshape}_{h_1 \times d_1}(\boldsymbol{W}_r^{A^Z} \boldsymbol{x}_t), \qquad \boldsymbol{B}^Z_{t,r} = \text{reshape}_{h_2 \times d_2}(\boldsymbol{W}_r^{B^Z} \boldsymbol{x}_t), \tag{2}$$

where $\boldsymbol{W}_r^{A^Z} \in \mathbb{R}^{h_1 d_1 \times d_{\text{model}}}$ and $\boldsymbol{W}_r^{B^Z} \in \mathbb{R}^{h_2 d_2 \times d_{\text{model}}}$ are learned projections and reshape fills rows in order. The query, key, and value matrices are the normalized sums

$$\boldsymbol{Q}_t = \frac{1}{R_Q} \sum_{r=1}^{R_Q} \boldsymbol{A}^Q_{t,r} \otimes \boldsymbol{B}^Q_{t,r}, \tag{3}$$

$$\boldsymbol{K}_t = \frac{1}{R_K} \sum_{r=1}^{R_K} \boldsymbol{A}^K_{t,r} \otimes \boldsymbol{B}^K_{t,r}, \tag{4}$$

$$\boldsymbol{V}_t = \frac{1}{R_V} \sum_{r=1}^{R_V} \boldsymbol{A}^V_{t,r} \otimes \boldsymbol{B}^V_{t,r}, \qquad \boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t \in \mathbb{R}^{h \times d_h}. \tag{5}$$

We refer to the number of terms R_Z as the rank of each representation. In practice, the R_Z projections in equation 2 are stacked into a single matrix and reshaped to $R_Z \times h_1 \times d_1$ and $R_Z \times h_2 \times d_2$ tensors. For later use, we write the rows of the factors as column vectors, $\boldsymbol{a}^Z_{t,r,i_1} = (\boldsymbol{A}^Z_{t,r}[i_1, :])^\top \in \mathbb{R}^{d_1}$ and $\boldsymbol{b}^Z_{t,r,i_2} = (\boldsymbol{B}^Z_{t,r}[i_2, :])^\top \in \mathbb{R}^{d_2}$.

Relation to TPA. KPA contains TPA (Zhang et al., 2025) as a special case: the split $(h_1, d_1, h_2, d_2) = (h, 1, 1, d_h)$ reduces each Kronecker term to the outer product of a head vector and a feature vector. In TPA, and in its higher-order extension (Zhang et al., 2025, Appendix C), each term gives every head the same feature vector, multiplied by a different number per head. As a result, with R terms, the vectors of all heads are combinations of the same R feature vectors. In KPA, each factor pairs a head sub-index with a feature sub-index (Table 1), so heads that use different rows of $\boldsymbol{A}$ or $\boldsymbol{B}$ can receive feature vectors with different directions within a single term. A single term has matrix rank $\text{rank}(\boldsymbol{A})\,\text{rank}(\boldsymbol{B})$, which can be as large as $\min(h_1, d_1)\min(h_2, d_2)$, and each head computes its scores from its own factor inner products (Proposition 3.2). The factor split therefore controls how much the heads share their feature directions, with TPA as the endpoint in which all heads share them.

3.2 Factored Scores and Attention Output

Applying the mixed-product identity in equation 1 to each head's query–key inner product gives an exact contraction.

Proposition 3.2 (Factored score)**.** *Before positional rotation, the unscaled score at head $i = (i_1, i_2)$ is*

$$\text{score}_i(t, s) = \frac{1}{R_Q R_K} \sum_{r=1}^{R_Q} \sum_{u=1}^{R_K} \big((\boldsymbol{a}^Q_{t,r,i_1})^\top \boldsymbol{a}^K_{s,u,i_1}\big)\big((\boldsymbol{b}^Q_{t,r,i_2})^\top \boldsymbol{b}^K_{s,u,i_2}\big). \tag{6}$$

The proof is in Appendix A. Each score is thus built from a d_1-dimensional and a d_2-dimensional inner product, and each of these is shared by h_2 and h_1 heads, respectively. Let $\ell_i(t,s)$ denote the score after the rotary encoding, given by equation 9 in Section 3.3. Causal attention uses

$$\alpha_{t,s,i} = \frac{\exp(\ell_i(t,s)/\sqrt{d_h})}{\sum_{v=1}^{t}\exp(\ell_i(t,v)/\sqrt{d_h})} \quad (s \le t), \qquad \alpha_{t,s,i} = 0 \quad (s > t), \tag{7}$$

and the output of head i and of the attention block are

$$\boldsymbol{z}_{t,i} = \frac{1}{R_V}\sum_{s=1}^{t}\sum_{r=1}^{R_V}\alpha_{t,s,i}\big(\boldsymbol{a}^V_{s,r,i_1}\otimes\boldsymbol{b}^V_{s,r,i_2}\big), \qquad \boldsymbol{y}_t = \boldsymbol{W}^O\,\mathrm{concat}_{i=1}^{h}\,\boldsymbol{z}_{t,i}, \tag{8}$$

where $\boldsymbol{z}_{t,i}\in\mathbb{R}^{d_h}$ and $\boldsymbol{W}^O\in\mathbb{R}^{d_{\text{model}}\times hd_h}$. These equations are exactly equivalent to reconstructing $\boldsymbol{Q}_t$, $\boldsymbol{K}_t$, and $\boldsymbol{V}_t$ and applying standard attention, but can be evaluated directly from the cached factors (Appendix A.4).

3.3 Kronecker-Separable Rotary Encoding

Definition 3.3 (Kronecker-separable RoPE)**.** *Assume d_2 is even. Let $\boldsymbol{R}^{(2)}_\theta(t)$ be the block-diagonal d_2-dimensional rotation with two-dimensional blocks of angle $t\theta_j$, where $\theta_j = 10000^{-2j/d_2}$ for $j = 0,\dots,d_2/2-1$. With the feature ordering specified above, define $\boldsymbol{R}_{\mathrm{KPA}}(t) = \boldsymbol{I}_{d_1}\otimes\boldsymbol{R}^{(2)}_\theta(t)$.*

This repeats the same d_2-dimensional frequency schedule across d_1 blocks. For $d_1 > 1$, it is a modified rotary operator, rather than the usual full-d_h frequency schedule of RoPE (Su et al., 2024). This separable construction has an antecedent in higher-order TPA (Zhang et al., 2025, Appendix C.1).

Theorem 3.4 (Factor compatibility of the rotary operator)**.** *For $Z\in\{Q,K\}$, rotating head vector $R_Z^{-1}\sum_r \boldsymbol{a}^Z_{t,r,i_1}\otimes\boldsymbol{b}^Z_{t,r,i_2}$ by $\boldsymbol{R}_{\mathrm{KPA}}(t)$ is equivalent to replacing each $\boldsymbol{b}^Z_{t,r,i_2}$ by $\boldsymbol{R}^{(2)}_\theta(t)\boldsymbol{b}^Z_{t,r,i_2}$. The resulting unscaled score is*

$$\ell_i(t,s) = \frac{1}{R_Q R_K}\sum_{r=1}^{R_Q}\sum_{u=1}^{R_K}\big((\boldsymbol{a}^Q_{t,r,i_1})^\top\boldsymbol{a}^K_{s,u,i_1}\big)\big((\boldsymbol{b}^Q_{t,r,i_2})^\top\boldsymbol{R}^{(2)}_\theta(s-t)\boldsymbol{b}^K_{s,u,i_2}\big). \tag{9}$$

Appendix A gives the proof. Only the *explicit rotary* dependence reduces to $s-t$; the contextual factors still depend separately on $\boldsymbol{x}_t$ and $\boldsymbol{x}_s$. In matrix form, the rotated factors are $\widetilde{\boldsymbol{B}}^Z_{t,r} = \boldsymbol{B}^Z_{t,r}\boldsymbol{R}^{(2)}_\theta(t)^\top$. Thus keys may be cached as $(\boldsymbol{A}^K_{t,r},\widetilde{\boldsymbol{B}}^K_{t,r})$. The $\boldsymbol{A}^K$ factors are unrotated, not position-independent. Values remain unrotated. This operator preserves the factor shapes and needs no additional rotary-key cache branch.

3.4 Cache, Parameters, and Computation

During decoding, each layer caches, for every past token, the key factors $\{\boldsymbol{A}^K_{t,r},\widetilde{\boldsymbol{B}}^K_{t,r}\}_{r=1}^{R_K}$ and the value factors $\{\boldsymbol{A}^V_{t,r},\boldsymbol{B}^V_{t,r}\}_{r=1}^{R_V}$. Each term stores $h_1d_1 + h_2d_2$ numbers. We denote by N_{KV} the number of scalars cached per token and layer, and by C_{KPA} the compression ratio, defined as the number of scalars cached by MHA with the same h and d_h, namely $2hd_h$, divided by N_{KV}:

$$N_{\mathrm{KV}} = (R_K+R_V)(h_1d_1+h_2d_2), \qquad C_{\mathrm{KPA}} = \frac{2hd_h}{N_{\mathrm{KV}}}. \tag{10}$$

For example, with $h=16$, $d_h=64$, a (4×4) head split, $d_1=d_2=8$, and $R_K=R_V=2$, we have $N_{\mathrm{KV}}=256$, compared with 2048 for MHA, so $C_{\mathrm{KPA}}=8$. Because the product $(h_1d_1)(h_2d_2) = hd_h$ is fixed, the arithmetic–geometric mean inequality implies that the sum $h_1d_1+h_2d_2$, and hence N_{KV}, is smallest when the two factors have equal size. Balanced splits therefore give the highest compression, subject to integer factorisations and an even d_2.

The factor projections contain $d_{\text{model}}(R_Q+R_K+R_V)(h_1d_1+h_2d_2)$ parameters, excluding the output projection and biases. Since queries are not cached, R_Q can be increased without affecting N_{KV}, at the cost of more query parameters and more score computation, which sums over R_QR_K factor pairs. These counts describe storage, not measured memory use or latency; Appendix A.4 gives an exact decoding procedure from the cached factors.

Table 2: Zero-shot scores (%) at the 353M scale. Baselines are imported from Zhang et al. (2025); see Section 4 for comparison scope. Avg. is the unweighted mean recomputed from the nine displayed scores. Column maxima, including ties, are **bold**. HellaSw. = HellaSwag; W.G. = WinoGrande.

Method	ARC-E	ARC-C	BoolQ	HellaSw.	OBQA	PIQA	W.G.	MMLU	SciQ	Avg.
MHA	59.51	29.52	59.60	45.68	34.20	68.82	53.43	23.33	76.90	50.11
MQA	57.62	31.91	59.45	45.69	35.40	69.31	53.51	26.47	74.60	50.44
GQA	58.67	31.48	58.29	45.45	35.20	68.50	54.46	24.58	76.50	50.35
MLA	56.65	29.52	57.83	46.05	34.60	69.42	52.80	24.62	79.70	50.13
TPA	58.38	31.57	59.39	**46.83**	**37.00**	70.02	54.06	25.52	79.90	51.41
KPA h=(4×4)	64.23	30.72	59.66	44.09	34.40	70.08	54.38	**26.79**	**85.30**	52.18
KPA h=(2×8)	64.15	31.48	57.21	45.93	36.20	70.18	**54.92**	26.31	83.90	52.25
KPA h=(8×2)	**66.16**	**33.02**	**60.37**	45.11	35.80	**71.29**	53.12	**26.79**	84.40	**52.90**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We train KPA models from scratch at four scales, 125M, 353M, 773M, and 1.5B parameters, following the protocol of Zhang et al. (2025). All models use a Llama-style architecture built on nanoGPT (Karpathy, 2022) with SwiGLU activations and are trained on FineWeb-Edu-100B (Penedo et al., 2024) for 50B tokens, matching the training budget of the TPA baselines. KPA uses the Kronecker-separable rotary operator of Section 3.3, with $d_1 = d_2 = 8$, $R_K = R_V = 2$, and $R_Q = 6$. We evaluate zero-shot on nine benchmarks with `lm-evaluation-harness` (Gao et al., 2023). Scores for MHA, MQA, GQA, MLA, and TPA are taken from Zhang et al. (2025). Appendix C gives the full training and evaluation details.

Each reported average is the unweighted mean of the nine task scores, and differences are given in percentage points (pp). Cache ratios are analytical counts of stored KV elements relative to MHA with the same number of heads and head width (Section 3.4 and Appendix B).

4.2 MAIN RESULTS

Tables 2 and 3 show the 353M and 773M results. At 353M, KPA $h = (8 \times 2)$ has the highest average, 52.90, versus 51.41 for TPA, a difference of 1.49 pp. Its KV compression is $6.4\times$. KPA $h = (4 \times 4)$ reaches an average of 52.18 with $8.0\times$ compression. At 773M, KPA $h = (10 \times 2)$ reaches 55.13 versus 53.14 for TPA, a difference of 1.99 pp, with $6.7\times$ compression. The more balanced $h = (4 \times 5)$ configuration reaches 54.55 with $8.9\times$ compression.

The improvements are not uniform across tasks. At 353M, the highest-average KPA configuration scores below TPA on HellaSwag, OpenbookQA, and WinoGrande; at 773M, it scores below TPA on HellaSwag and MMLU. Appendix D reports the 125M and 1.5B results, whose highest KPA averages are 47.06 and 56.55, respectively. At all four scales, the best KPA configuration has the highest average of all methods.

4.3 ACCURACY AND STORAGE ACROSS HEAD SPLITS

The evaluated configurations vary $h_1 \times h_2 = h$, with the same reported feature split $d_1 = d_2 = 8$ and KV ranks $R_K = R_V = 2$. At 353M, reversing (2×8) to (8×2) increases the reported average from 52.25 to 52.90 while leaving the analytical KV count unchanged. At 773M, reversing (2×10) to (10×2) changes 53.80 to 55.13, also without changing that count. Because RoPE acts on the second factor, swapping the head split changes which factor receives more head capacity. This offers a possible explanation for the observed pattern, which we leave to future work to test directly.

For fixed h and $d_1 = d_2$, more balanced head splits reduce the sum of factor sizes and therefore the analytical cache count. In these tables, the balanced or near-balanced configurations use fewer KV elements than the highest-average configurations. The head split therefore lets practitioners choose between higher accuracy and a smaller cache.

Table 3: Zero-shot scores (%) at the 773M scale. Baselines are imported from Zhang et al. (2025); see Section 4 for comparison scope. Avg. is the unweighted mean recomputed from the nine displayed scores. Column maxima, including ties, are **bold**. HellaSw. = HellaSwag; W.G. = WinoGrande.

Method	ARC-E	ARC-C	BoolQ	HellaSw.	OBQA	PIQA	W.G.	MMLU	SciQ	Avg.
MHA	59.93	33.62	61.93	50.63	36.00	71.06	55.41	22.87	81.20	52.52
MQA	60.73	33.62	57.34	50.09	37.00	69.97	55.49	**25.30**	79.60	52.13
GQA	61.66	34.30	58.72	49.85	38.40	71.16	53.75	25.23	77.60	52.30
MLA	63.55	32.85	60.95	**51.72**	38.80	70.51	55.01	24.55	81.90	53.32
TPA	63.22	35.58	60.03	51.26	36.80	71.44	55.56	24.77	79.60	53.14
KPA h=(4×5)	68.56	35.92	60.09	50.28	36.80	**72.85**	55.33	25.26	85.90	54.55
KPA h=(2×10)	68.31	34.98	55.32	50.91	38.00	71.06	55.64	24.66	85.30	53.80
KPA h=(10×2)	**68.89**	**36.01**	**62.03**	50.38	**39.40**	**72.85**	**55.98**	24.52	**86.00**	**55.13**

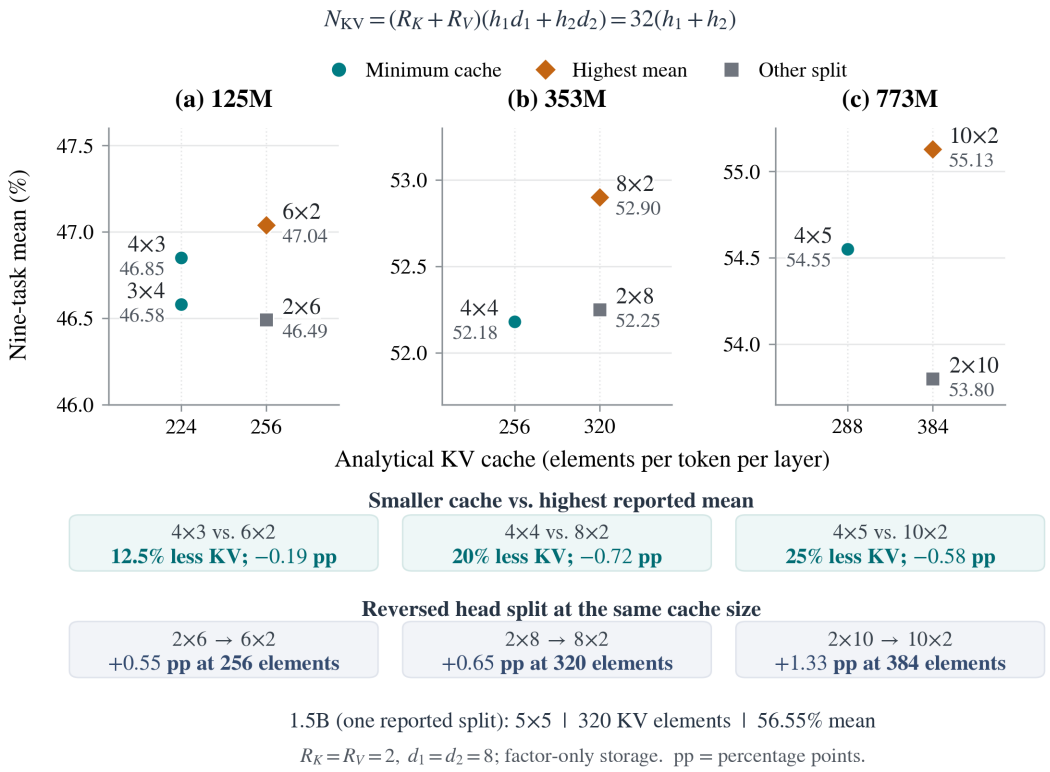

Figure 2: **Accuracy and KV storage of KPA configurations** at the 125M, 353M, and 773M scales. Each point is one configuration, labeled by its head split $h_1 \times h_2$ and nine-task average accuracy (%). The upper row compares the best configuration at the smallest cache size with the best configuration overall at each scale; storage savings are relative to the latter. The lower row compares reversed head splits, which have equal cache size. Storage counts assume $R_K = R_V = 2$, $d_1 = d_2 = 8$, and caching only the key and value factors.

Figure 2 summarizes the reported KPA configurations. At 125M, 353M, and 773M, the highest-scoring configuration at the smaller cache size uses 12.5%, 20%, and 25% fewer KV elements than the highest-average configuration at the same nominal scale, with respective reductions of 0.19, 0.72, and 0.58 percentage points in the average accuracy. Reversing the asymmetric head split from smaller to larger h_1 is associated with differences of $+0.55$, $+0.65$, and $+1.33$ percentage points, respectively, at unchanged analytical cache size.

5 Conclusion

KPA makes the grouping of head and feature indices an explicit design choice for compact attention representations. Its contextual matrix factors couple a head sub-index with a feature sub-index, yielding a Kronecker structure characterized by rank under rearrangement. This structure contains TPA as a special case, while non-degenerate splits let different heads receive different feature directions within a single term. Exact score and value contractions, together with a specified factor-separable rotary operator, describe how attention can operate on this representation while retaining only compact key and value factors, without a decoupled rotary branch.

Across the four model scales, the best KPA configuration has a higher nine-task average than all baselines. The evaluated factor shapes require 6–$10\times$ fewer persistent KV entries than same-shape MHA under factor-only caching. The head-split results expose distinct accuracy and storage operating points: balanced factor element counts minimize storage, whereas asymmetric splits can yield higher observed scores. These findings motivate further study of factor grouping as an attention design choice. Establishing practical serving benefits requires an efficient factored decoding kernel and measured memory and latency, particularly on long contexts and memory-constrained deployments, where cache compression matters most.

Reproducibility Statement

Section 3 specifies the factor shapes, attention equations, rotary operator, and cache-counting convention. Appendix A gives the proofs and an exact decoding procedure, Appendix B gives storage counts for every KPA configuration, and Appendix C gives the training and evaluation settings. All averages are computed from the task scores reported in the tables. We will release code for KPA and the trained configurations upon publication.

Statement on the Use of AI Tools

OpenAI ChatGPT/Codex assisted with this revision through language editing and organization. This assistance did not generate model-training or benchmark observations. The authors are responsible for the correctness, originality, citations, and final content of the manuscript.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zelaski, Sumit Sanghai, and Yuhuai Fu. 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.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

Tri Dao, Beidi Chen, Nimit S. Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning (ICML)*, 2022.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. 2024.

Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for GPT compression. In *ACL*, 2022.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter-efficient tuning with kronecker adapter. In *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*, pp. 49–65. Springer, 2025.

Leo Gao, Jonathan Tow, Stella Biderman, et al. A framework for few-shot language model evaluation. `https://github.com/EleutherAI/lm-evaluation-harness`, 2023.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jingcheng Hu, Houyi Li, Yinmin Zhang, Zili Wang, Shuigeng Zhou, Xiangyu Zhang, and Heung Yeung Shum. Multi-matrix factorization attention. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25114–25126, 2025.

Andrej Karpathy. nanogpt. `https://github.com/karpathy/nanoGPT`, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning (ICML)*, 2024.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Soroush Omranpour, Guillaume Rabusseau, and Reihaneh Rabbany. Higher order transformers: Efficient attention mechanism for tensor structured data. 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. 2019.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kroneckerbert: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2116–2127, 2022.

Charles F. Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear Algebra for Large Scale and Real-Time Applications*, pp. 293–314. Springer, 1993.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Ted Zadouri, Hubert Strauss, and Tri Dao. Hardware-efficient attention for fast decoding. *arXiv preprint arXiv:2505.21487*, 2025.

Yifan Zhang, Yifeng Liu, et al. Tensor product attention is all you need. In *Advances in Neural Information Processing Systems*, 2025. Spotlight.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A Algebraic Details and Exact Factor-Based Attention

A.1 Rearrangement Rank

The number of Kronecker terms needed to represent a given matrix is characterised by the following classical result.

Proposition A.1 (Rank under rearrangement). *For $\boldsymbol{X} \in \mathbb{R}^{h \times d_h}$, define $\mathcal{R}(\boldsymbol{X}) \in \mathbb{R}^{h_1 d_1 \times h_2 d_2}$ by $\mathcal{R}(\boldsymbol{X})_{(i_1,j_1),(i_2,j_2)} = X_{(i_1,i_2),(j_1,j_2)}$. The smallest R for which $\boldsymbol{X} = \sum_{r=1}^{R} \boldsymbol{A}_r \otimes \boldsymbol{B}_r$ with arbitrary $\boldsymbol{A}_r \in \mathbb{R}^{h_1 \times d_1}$ and $\boldsymbol{B}_r \in \mathbb{R}^{h_2 \times d_2}$ is $\operatorname{rank}(\mathcal{R}(\boldsymbol{X}))$.*

Proof. Let $\mathrm{vec}_{\mathrm{row}}$ stack matrix rows into a column vector. With the ordering in Section 3.1, entrywise evaluation gives

$$\mathcal{R}(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathrm{vec}_{\mathrm{row}}(\boldsymbol{A}) \, \mathrm{vec}_{\mathrm{row}}(\boldsymbol{B})^{\top}. \tag{11}$$

Thus a sum of R Kronecker products rearranges to a sum of R rank-one matrices, implying $\operatorname{rank}(\mathcal{R}(\boldsymbol{X})) \le R$. Conversely, a rank-$q$ decomposition $\mathcal{R}(\boldsymbol{X}) = \sum_{r=1}^{q} \boldsymbol{u}_r \boldsymbol{v}_r^{\top}$ can be inverted by reshaping $\boldsymbol{u}_r$ to $h_1 \times d_1$ and $\boldsymbol{v}_r$ to $h_2 \times d_2$, which yields q Kronecker terms for $\boldsymbol{X}$. The normalization $1/R$ in KPA does not affect this result, since it can be absorbed into one factor of each term. This argument is the standard rearrangement construction of Van Loan & Pitsianis (1993). □

Ordinary matrix rank is a different constraint. For any matrices $\boldsymbol{A}$ and $\boldsymbol{B}$,

$$\operatorname{rank}(\boldsymbol{A} \otimes \boldsymbol{B}) = \operatorname{rank}(\boldsymbol{A}) \operatorname{rank}(\boldsymbol{B}).$$

For example, this follows by taking singular value decompositions: the nonzero singular values of their Kronecker product are all pairwise products of nonzero singular values. Hence a KPA slice satisfies

$$\operatorname{rank}(\boldsymbol{Z}_t) \le \min\{h, d_h, R_Z \min(h_1, d_1) \min(h_2, d_2)\}.$$

This upper bound is not a guarantee of higher capacity, and does not assert that all matrices satisfying it are representable with R_Z terms.

A small example demonstrates why there is no general containment claim at a fixed non-degenerate split. Set $h = d_h = 4$ and $h_1 = h_2 = d_1 = d_2 = 2$. The matrix $\boldsymbol{X} = \boldsymbol{I}_2 \otimes \boldsymbol{I}_2 = \boldsymbol{I}_4$ has Kronecker term count one but ordinary matrix rank four, so it cannot be represented by one ordinary outer product. In the other direction, let $\boldsymbol{u} = \mathrm{vec}_{\mathrm{row}}(\boldsymbol{I}_2) = (1, 0, 0, 1)^{\top}$ and $\boldsymbol{Y} = \boldsymbol{u}\boldsymbol{u}^{\top}$. Then $\boldsymbol{Y}$ has ordinary matrix rank one, while

$$\mathcal{R}(\boldsymbol{Y})_{(i_1,j_1),(i_2,j_2)} = \mathbf{1}_{i_1=i_2} \mathbf{1}_{j_1=j_2}, \qquad \mathcal{R}(\boldsymbol{Y}) = \boldsymbol{I}_4.$$

It therefore requires four Kronecker terms for this split. The one-term Kronecker and ordinary outer-product representations each use eight scalar factor entries in this example, yet neither represented matrix family contains the other. This is a pointwise comparison, not a comparison of optimisation behaviour or a claim about task performance. Likewise, the rearrangement-rank characterization allows arbitrary factors for a fixed matrix; it does not establish that the contextual linear maps realise every such factorization simultaneously across inputs.

Comparison with higher-order TPA. Under the feature ordering used here, a third-order TPA term from Zhang et al. (2025, Appendix C) has entries $a_{(i_1,i_2)} c_{j_1} b_{j_2}$, after identifying its two feature dimensions with d_1, d_2. A KPA term instead has entries $A_{i_1,j_1} B_{i_2,j_2}$. The former uses a vector over all heads and two feature vectors; the latter uses two matrices that each couple part of the head and feature indices. This is a difference in tensor grouping, not an ordering of their attainable quality at a fixed budget. The per-term factor counts are correspondingly $h + d_1 + d_2$ and $h_1 d_1 + h_2 d_2$ when all factors are cached. Higher-order TPA also provides a separable rotary construction and a small-model evaluation; neither factor-level rotation nor higher-order contextual factorization is unique to KPA.

A.2 Proof of Proposition 3.2

Proof. The column vectors obtained by transposing row $i = (i_1, i_2)$ of the query and key matrices are

$$\boldsymbol{q}_{t,i} = \frac{1}{R_Q} \sum_{r=1}^{R_Q} \boldsymbol{a}^{Q}_{t,r,i_1} \otimes \boldsymbol{b}^{Q}_{t,r,i_2}, \qquad \boldsymbol{k}_{s,i} = \frac{1}{R_K} \sum_{u=1}^{R_K} \boldsymbol{a}^{K}_{s,u,i_1} \otimes \boldsymbol{b}^{K}_{s,u,i_2}.$$

The mixed-product identity gives

$$(\boldsymbol{a} \otimes \boldsymbol{b})^\top (\boldsymbol{c} \otimes \boldsymbol{d}) = (\boldsymbol{a}^\top \boldsymbol{c})(\boldsymbol{b}^\top \boldsymbol{d}).$$

Expanding $\boldsymbol{q}_{t,i}^\top \boldsymbol{k}_{s,i}$ and applying this identity to each pair of terms yields equation 6, including its normalization $1/(R_Q R_K)$. The score is unscaled; the factor $1/\sqrt{d_h}$ enters the softmax in equation 7. □

A.3 Proof of Theorem 3.4

Proof. For column vectors of lengths d_1 and d_2, respectively,

$$(\boldsymbol{I}_{d_1} \otimes \boldsymbol{R}_\theta^{(2)}(t))(\boldsymbol{a} \otimes \boldsymbol{b}) = \boldsymbol{a} \otimes (\boldsymbol{R}_\theta^{(2)}(t)\boldsymbol{b}).$$

Linearity extends this identity to the normalized sum over factor terms. With the stated joint feature ordering, the matrix operation on the original row factors is therefore $\boldsymbol{B} \mapsto \boldsymbol{B}\boldsymbol{R}_\theta^{(2)}(t)^\top$. Each two-dimensional rotation block is orthogonal and obeys the rotation addition law, so

$$\boldsymbol{R}_\theta^{(2)}(t)^\top \boldsymbol{R}_\theta^{(2)}(s) = \boldsymbol{R}_\theta^{(2)}(s-t).$$

Apply this identity to the $\boldsymbol{b}$ inner product in every term of equation 6 to obtain equation 9. The result concerns the explicit positional operator, with the factors at t and s left intact. It does not remove their contextual dependence. □

Frequency schedule and layout. The theorem assumes the feature order (j_1, j_2) with j_2 varying fastest and adjacent coordinate pairs inside each d_2 block. A different layout requires the corresponding fixed permutation of both the vectors and the rotation operator. For $d_1 > 1$, the operator repeats the same frequencies in each block, whereas standard full-dimensional RoPE typically assigns frequencies across all $d_h/2$ coordinate pairs. For $d_1 = 1$, the construction reduces to the usual d_h-dimensional schedule. The same type of separable operator appears in the higher-order TPA construction of Zhang et al. (2025, Appendix C.1).

A.4 Exact Autoregressive Decoding from Cached Factors

The following specifies one mathematical execution of the model; it is not a report of a measured decoding kernel. At token position t in one layer:

1. Compute the query, key, and value factors from $\boldsymbol{x}_t$ using equation 2. Rotate the query and key $\boldsymbol{B}$ factors: $\widetilde{\boldsymbol{B}}_{t,r}^Z = \boldsymbol{B}_{t,r}^Z \boldsymbol{R}_\theta^{(2)}(t)^\top$ for $Z \in \{Q, K\}$. Do not rotate the value factors.
2. Append $(\boldsymbol{A}_{t,r}^K, \widetilde{\boldsymbol{B}}_{t,r}^K)$ and $(\boldsymbol{A}_{t,r}^V, \boldsymbol{B}_{t,r}^V)$ to the cache. The current query factors are temporary and need not persist after this token's computation.
3. For each cached token $s \le t$ and head (i_1, i_2), compute equation 6 using the rotated $\boldsymbol{b}^Q$ and cached rotated $\boldsymbol{b}^K$. This equals equation 9; do not apply a second relative rotation to already-rotated factors.
4. Apply the causal scaled softmax in equation 7. Accumulate the value-factor products with these weights using equation 8, then concatenate heads and apply $\boldsymbol{W}^O$.

The equality with reconstructed attention holds in exact arithmetic; floating-point contraction order can change rounding. A tiled or online softmax can change the schedule without changing the mathematical model. Whether a particular implementation achieves lower peak memory or latency requires an implementation-level evaluation.

Storage and arithmetic are distinct. For batch size B, L layers with identical configurations, context length T, and w bytes per cached scalar, the factor cache contains $BLTw(R_K + R_V)(h_1 d_1 + h_2 d_2)$ bytes. This excludes cache metadata, padding, position/frequency tables, projection weights, and workspace. The comparison with $2BLTwhd_h$ bytes for MHA assumes equal cache precision and head width. Neither value is a measurement of allocated device memory.

The factor projection parameter count in Section 3.4 follows by summing the numbers of outputs of the two linear maps across Q, K, V and multiplying by d_{model}. The shared output projection adds $d_{\text{model}} h d_h$ parameters. Bias terms, if present in an implementation, must be counted separately.

For one query–key position pair, a direct contraction can reuse the $\boldsymbol{a}$ inner products across i_2 and the $\boldsymbol{b}$ inner products across i_1. Computing these inner products and combining them for every head requires $O(R_Q R_K (h_1 d_1 + h_2 d_2 + h))$ arithmetic operations, excluding factor projections and rotary encoding. A straightforward value contraction requires $O(R_V h d_h)$ operations per attended token. These counts describe one possible evaluation order, not a lower bound or a measured speedup. They illustrate why small factor storage and efficient execution are separate questions, particularly as $R_Q R_K$ grows.

B Analytical KV Cache Accounting

The following counts assume that an implementation retains only KPA's key and value factor matrices for each past token. They describe persistent KV elements per token per layer, excluding query projections, model parameters, temporary reconstructed tensors, attention workspaces, allocator overhead, and implementation-specific state. Let $h = h_1 h_2$, $d_h = d_1 d_2$, and let b be the number of bytes used to store one element. The KPA and same-shape MHA reference counts are

$$N_{\text{KPA}} = (R_K + R_V)(h_1 d_1 + h_2 d_2), \tag{12}$$

$$N_{\text{MHA}} = 2 h_1 h_2 d_1 d_2. \tag{13}$$

Their ratio is

$$C_{\text{KPA}} = \frac{N_{\text{MHA}}}{N_{\text{KPA}}} = \frac{2 h_1 h_2 d_1 d_2}{(R_K + R_V)(h_1 d_1 + h_2 d_2)}. \tag{14}$$

At a common storage precision, the factor b cancels from this ratio. For batch size B, cached length T, and L layers with the same configuration, the corresponding idealized storage is $BTLbN$ bytes. For example, the 353M (4×4) configuration stores 256 elements per token per layer, or 512 bytes at two bytes per element, versus 4096 bytes for same-shape MHA.

Table 4 lists all eleven evaluated KPA head splits, with $R_K = R_V = 2$ and $d1 = d2 = 8$. We do not report a TPA ratio, since it depends on TPA's head count, feature width, and ranks rather than on model scale alone.

Table 4: Analytical persistent KV elements per token per layer, assuming $R_K = R_V = 2$ and $d_1 = d_2 = 8$. MHA uses the same $h = h_1 h_2$ and $d_h = 64$ as each KPA row. Ratios are rounded to one decimal place; counts are exact under the stated assumptions.

Scale	Head split	KPA elements	MHA elements	Ratio
125M	3×4	224	1536	6.9×
125M	4×3	224	1536	6.9×
125M	2×6	256	1536	6.0×
125M	6×2	256	1536	6.0×
353M	4×4	256	2048	8.0×
353M	2×8	320	2048	6.4×
353M	8×2	320	2048	6.4×
773M	4×5	288	2560	8.9×
773M	2×10	384	2560	6.7×
773M	10×2	384	2560	6.7×
1.5B	5×5	320	3200	10.0×

For fixed h and $d_1 = d_2$, the cache count is proportional to $h_1 + h_2$. Balanced factors minimize this sum subject to the integer factorization of h. Reversing h_1 and h_2 leaves the count unchanged, even though the parameterization and reported task scores can differ. Accordingly, the (4×4) and (4×5) configurations have lower analytical storage than their more asymmetric counterparts at the same respective model scale.

C Training and Evaluation Details

Architecture and data. All KPA models are built on nanoGPT (Karpathy, 2022) with a Llama-style architecture, SwiGLU activations (Shazeer, 2020), and the Kronecker-separable rotary operator of Section 3.3. We use $d_1 = d_2 = 8$, $R_K = R_V = 2$, and $R_Q = 6$, with the head splits listed in Table 4. Models are trained from scratch on FineWeb-Edu-100B (Penedo et al., 2024) for 50,000 iterations of approximately one million tokens each, for a total of approximately 50B tokens, matching the training budget of the baselines in Zhang et al. (2025).

Optimization. We use AdamW with a peak learning rate of 6×10^{-4} at 125M and 353M and 3×10^{-4} at 773M and 1.5B, 2,000 warmup steps, cosine decay to one-tenth of the peak, and weight decay 0.1. Training uses bfloat16 precision on NVIDIA H100 GPUs.

Evaluation. We evaluate zero-shot with `lm-evaluation-harness` (Gao et al., 2023) on ARC-Easy, ARC-Challenge, BoolQ, HellaSwag, OpenbookQA, PIQA, WinoGrande, MMLU, and SciQ. Following Zhang et al. (2025), we report normalized accuracy for ARC-Easy, ARC-Challenge, HellaSwag, OpenbookQA, PIQA, and SciQ, and standard accuracy for BoolQ, WinoGrande, and MMLU. Baseline scores for MHA, MQA, GQA, MLA, and TPA are taken from Zhang et al. (2025). Each reported average is the unweighted mean of the nine task scores, rounded to two decimal places.

D Additional Experimental Results

D.1 1.5B Scale

Table 5 reports the evaluated $h_1 = h_2 = 5$ configuration. For $h = 25$, this is the only integer head split with both factors greater than one; the degenerate (1×25) and (25×1) splits are not evaluated. Its recomputed average is 56.55, compared with 55.01 for TPA (a difference of 1.54 pp). KPA has the highest reported score in eight of the nine task columns. On MMLU, KPA scores 24.38, below MHA's 25.44 and TPA's 24.49. The corresponding analytical KV compression is $10.0\times$ relative to same-shape MHA. A single evaluated head split at this scale does not establish how other splits would behave.

D.2 125M Scale

Table 6 reports four KPA configurations. Their recomputed averages range from 46.49 to 47.06, compared with 46.21 for TPA. KPA $h = (6 \times 2)$ has the highest average, 47.06, a difference of 0.83 pp from TPA. These average-score gains do not imply gains on every task: that configuration scores below TPA on HellaSwag, OpenbookQA, and MMLU, and below MHA on BoolQ and WinoGrande. Reversing the head split improves the average for both the (3×4) versus (4×3) and (2×6) versus (6×2) pairs. As with the main results, these observations are descriptive and do not identify a causal explanation or establish significance.

Table 5: Zero-shot scores (%) at the 1.5B scale. Baselines are imported from Zhang et al. (2025); see Section 4 for comparison scope. Avg. is the unweighted mean recomputed from the nine displayed scores. Column maxima, including ties, are **bold**. HellaSw. = HellaSwag; W.G. = WinoGrande.

Method	ARC-E	ARC-C	BoolQ	HellaSw.	OBQA	PIQA	W.G.	MMLU	SciQ	Avg.
MHA	64.81	35.41	61.90	54.32	37.20	72.74	55.80	**25.44**	82.80	54.49
MQA	64.10	36.01	62.26	54.38	39.00	72.58	56.43	23.70	81.90	54.48
GQA	63.68	35.92	60.46	54.17	38.40	73.56	56.27	24.77	81.70	54.33
MLA	64.14	35.92	60.12	53.60	39.20	72.25	55.17	24.71	81.60	54.08
TPA	66.71	36.52	61.38	54.03	40.40	72.52	56.83	24.49	82.20	55.01
KPA h=(5×5)	**70.62**	**38.74**	**63.26**	**54.60**	**41.00**	**73.72**	**57.20**	24.38	**85.50**	**56.55**

Table 6: Zero-shot scores (%) at the 125M scale. Baselines are imported from Zhang et al. (2025); see Section 4 for comparison scope. Avg. is the unweighted mean recomputed from the nine displayed scores. Column maxima, including ties, are **bold**. HellaSw. = HellaSwag; W.G. = WinoGrande.

Method	ARC-E	ARC-C	BoolQ	HellaSw.	OBQA	PIQA	W.G.	MMLU	SciQ	Avg.
MHA	50.63	26.96	**59.39**	36.18	32.00	64.96	**51.85**	23.40	70.30	46.19
MQA	49.62	25.34	55.72	35.94	31.40	64.85	51.30	23.37	68.70	45.14
GQA	48.70	25.68	56.15	35.58	31.40	64.91	51.62	23.12	68.20	45.04
MLA	50.21	26.71	58.01	36.25	**32.80**	64.69	50.59	**24.67**	71.90	46.20
TPA	51.26	27.39	57.00	**36.68**	**32.80**	64.47	49.72	24.61	72.00	46.21
KPA $h{=}(3{\times}4)$	52.10	27.30	57.58	35.90	32.20	65.10	50.80	24.50	73.80	46.58
KPA $h{=}(4{\times}3)$	52.89	27.52	57.84	36.06	32.40	65.30	51.10	24.55	74.20	46.87
KPA $h{=}(2{\times}6)$	52.30	27.20	57.22	35.78	32.20	65.09	50.90	24.40	73.50	46.49
KPA $h{=}(6{\times}2)$	**53.51**	**27.66**	57.90	35.70	32.60	**65.60**	51.20	24.59	**74.80**	**47.06**