

Kronecker Product Attention: Memory-Efficient KV Compression via Structured Factorization

Bahareh Nikpour¹ Hossein Rajabzadeh^{2*} Maryam Dialameh^{2*}
Vahid Partovi Nia³ Parsa Omid¹

¹Huawei Technologies, Toronto, Canada ²University of Waterloo, Waterloo, Canada

³Huawei Noah's Ark Lab

bahar.nikpour@h-partners.com, {hossein.rajabzadeh, maryam.dialameh}@uwaterloo.ca,
{vahid.partovinia, parsa.omidi}@huawei.com

Abstract

Efficient autoregressive generation is increasingly limited by the memory footprint of the key-value (KV) cache, which grows linearly with sequence length, batch size, and model depth. Existing approaches such as Multi-Query Attention (MQA), Grouped-Query Attention (GQA), Multi-Head Latent Attention (MLA), and Tensor Product Attention (TPA) reduce this cost through different trade-offs, but often sacrifice expressivity, require additional positional-encoding machinery, or impose restrictive low-rank structure. We propose Kronecker Product Attention (KPA), which represents key and value activations as sums of Kronecker products of compact, token-dependent matrix factors. KPA strictly generalizes TPA: rank-one vector-outer-product factorization arises as a degenerate Kronecker split, while non-degenerate matrix factors can represent richer block-structured patterns. We further show that a structured rotary embedding operator matched to the Kronecker representation preserves the relative-position property of RoPE while rotating only the compact factors, removing the need for a decoupled RoPE branch as in MLA. Empirically, KPA matches or exceeds TPA, MHA, MQA, GQA, and MLA across four model scales and nine downstream benchmarks under identical training conditions, while achieving up to $10\times$ KV cache compression. The Kronecker head split is a tuneable hyperparameter that allows practitioners to navigate the compression-accuracy trade-off at design time without architectural changes.

1 Introduction

The deployment of large-scale Transformer-based language models is increasingly bottlenecked by inference-time memory rather than parameter count. At the core of this bottleneck is the key-value (KV) cache, which stores projected keys and values across every layer and token so they do not need to be recomputed during decoding. This cache grows linearly with sequence length, batch size, and head dimensionality. As context windows extend to hundreds of thousands of tokens and batch sizes increase for throughput, the KV cache can dominate total GPU memory consumption, leaving little headroom for model weights or activations and directly limiting the context length, batch size, and deployment scale that practitioners can afford, motivating a broad line of work on KV compression.

Existing approaches address this along several lines. Token eviction and sparse attention methods risk discarding relevant tokens [Zhang et al., 2023, Xiao et al., 2023]. Others propose off-chip storage of key-value states, trading memory pressure for increased I/O latency. At the architectural

*This work was done while the author was affiliated with Huawei Canada.

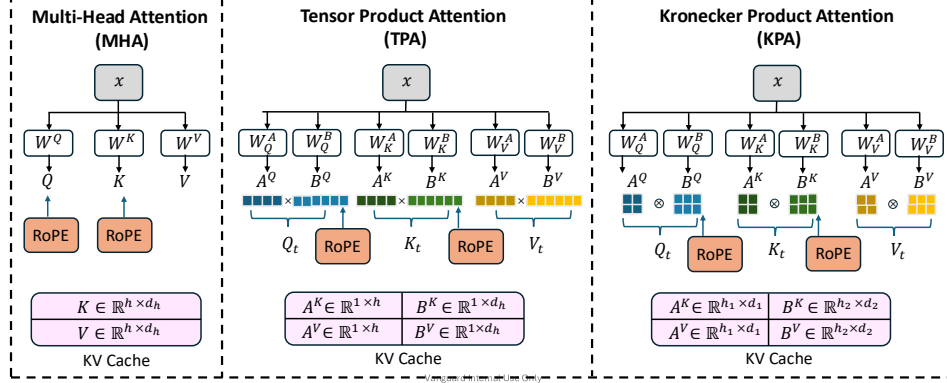


Figure 1: Overview of Multi-Head Attention (MHA), Tensor Product Attention (TPA), and Kronecker Product Attention (KPA, ours). MHA projects x directly into dense Q, K, V . TPA and KPA instead factor each projection into two components, combined via an outer product ($\times$) for TPA or a Kronecker product ($\otimes$) for KPA. RoPE is applied only to Q and K (never V), and rotates only the B factor in both methods (Theorem 2.4 formalizes this for KPA); A is cached unrotated. Since TPA’s A, B are vectors, their outer product is a flat, unstructured matrix; since KPA’s A, B are matrices, their Kronecker product is block-structured with a two-level head/feature hierarchy (shown by the internal grid lines). Only K, V factors enter the KV cache; Q is not cached. The token index t is omitted throughout except on the final Q_t, K_t, V_t , for clarity.

level, Multi-Query Attention (MQA) Shazeer [2019] and Grouped-Query Attention (GQA) [Ainslie et al., 2023] share keys and values across heads at the cost of representational diversity. Multi-Head Latent Attention (MLA) [DeepSeek-AI, 2024] avoids this by caching compressed latent representations instead, but requires a decoupled RoPE branch, in which a small set of uncompressed key dimensions carry the rotary embedding and are concatenated to the latent key at each decoding step. Tensor Product Attention (TPA) imposes low-rank CP (Canonical Polyadic) structure directly on KV projections, storing only the low-dimensional factors [Zhang et al., 2025]. However, CP decomposition operates on vectors and cannot jointly encode structure at both the head level and the feature level within each head. This limitation is important as empirical studies have shown that attention heads specialize into distinct functional roles, with different subsets attending to syntactic, semantic, and positional patterns, suggesting that an effective KV factorization should operate at both levels simultaneously [Michel et al., 2019, Voita et al., 2019].

Motivated by this, we use Kronecker product structure to represent per-token key and value matrices via two smaller contextual matrix factors. Unlike CP decomposition which operates on vectors, the Kronecker product expands two small matrices into a larger block-structured matrix with the same shape as a standard key or value slice, making it naturally suited to multi-head attention where both the head axis and the within-head feature axis have product structure. Among structured factorizations, the Kronecker product is distinguished by combining three properties in a single operation: its mixed-product identity lets inner products between Kronecker-structured queries and keys factor into products of smaller inner products; it admits a Kronecker-separable rotary operator that acts on only one compact factor; and a rank-1 outer product, which is the basic building block of CP decomposition, is a special case, making Kronecker factorization strictly more general than CP at the same parameter budget. We note that Kronecker structure has also been explored in attention mechanisms for encoder models operating on multi-dimensional inputs [Omranpour et al., 2024], reducing training-time computation rather than KV cache size. That setting differs fundamentally from autoregressive decoding, which is our focus.

We introduce **Kronecker Product Attention (KPA)**, a structured attention mechanism that applies this factorization to the query, key, and value projections of each attention head. This yields a family of KV compression schemes parameterized by the factor dimensions, allowing flexible trade-offs between compression ratio, head structure, and representational capacity. We provide a theoretical analysis characterizing the expressivity of KPA as a function of rank and factor dimensions, and prove that a modified frequency schedule we call Kronecker-Separable RoPE integrates seamlessly with the factorized KV representation without materializing the full key or value tensors and without

breaking the standard relative-position property. To evaluate KPA empirically, we train models at four scales including 124M, 353M, 773M, and 1.5B parameters on the FineWeb-Edu corpus. We compare KPA variants against TPA at matched compression ratios, finding that KPA matches or exceeds TPA across all scales. Figure 1 summarizes the contrast between MHA, TPA, and KPA. Our contributions are as follows:

- We propose Kronecker Product Attention (KPA), the first application of Kronecker product factorization to *per-token*, *contextual* key and value representations for KV cache compression in autoregressive decoding. Unlike prior Kronecker attention work that decomposes static attention score matrices for encoder models, KPA operates on input-dependent factors and directly reduces inference-time memory.
- We introduce Kronecker-Separable RoPE, which applies rotary position encoding directly to the compact key factors, preserving the standard relative-position property without a decoupled RoPE branch as in MLA.
- We train KPA at four scales (125M–1.5B) on FineWeb-Edu under identical conditions to TPA, showing KPA matches or exceeds TPA, MHA, MQA, GQA, and MLA across nine benchmarks while achieving $6.4 \times 10 \times$ KV cache compression.

2 Kronecker Product Attention

The key-value cache in standard multi-head attention stores, for each token t , a pair of matrices $\mathbf{K}_t, \mathbf{V}_t \in \mathbb{R}^{h \times d_h}$, whose entries are dense and unstructured. We argue that this is wasteful: attention heads are known to specialise into distinct functional roles [Michel et al., 2019, Voita et al., 2019], suggesting that the *head-level* dimension of $\mathbf{K}_t$ and $\mathbf{V}_t$ (i.e., how information is shared or differs across heads) carries structured, differentiable information. We hypothesise that the *feature-level* dimension (variation across features within a single head) benefits from an analogous structure, and design KPA. The proposed KPA makes this two-level structure explicit by factorising each key and value matrix as a sum of Kronecker products of two smaller matrices, one operating on a coarse partition of the head space, the other on the per-head feature space (formalised in Section 2.2), and caching only these factors. The result is a compression scheme that is structurally aligned with the functional organisation of multi-head attention in a way that CP decomposition, which operates on vectors rather than matrices, cannot achieve.

2.1 Background and Notation

Notation. Bold uppercase denotes matrices ($\mathbf{K}_t, \mathbf{V}_t, \mathbf{Q}_t, \mathbf{A}, \mathbf{B}, \mathbf{W}^K$); bold lowercase denotes vectors ($\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t, \mathbf{x}_t$); unbolded lowercase denotes scalars and indices ($h, d_h, d_{\text{model}}, R_K, t, i$); and calligraphic uppercase denotes higher-order tensors ($\mathcal{K}, \mathcal{V}, \mathcal{Q}$). We write $\otimes$ for the Kronecker product, $\mathbf{I}_n$ for the $n \times n$ identity, and $\mathbf{R}_\theta(t)$ for the RoPE rotation at position t . Subscripts index sequence positions; superscripts identify projections. We set $[n] := \{1, \dots, n\}$, and all vectors are column vectors unless stated otherwise.

Definition 2.1 (Kronecker product). *For matrices $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$, the Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$ is defined as*

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

The central algebraic property used in KPA is the mixed-product identity [Van Loan and Pitsianis, 1993] which states for conformable matrices $\mathbf{M}, \mathbf{N}, \mathbf{P}, \mathbf{Q}$, we can write:

$$(\mathbf{M} \otimes \mathbf{N})(\mathbf{P} \otimes \mathbf{Q}) = (\mathbf{MP}) \otimes (\mathbf{NQ}). \quad (1)$$

This identity underlies both the factored score computation (Section 2.3) and the RoPE compatibility proof (Section 2.4).

2.2 Factorised Key, Value, and Query Representations

Let $h = h_1 h_2$ and $d_h = d_1 d_2$, where h_1, h_2, d_1, d_2 are hyperparameters that control the two-level structure. For each token t with hidden state $\mathbf{x}_t \in \mathbb{R}^{d_{\text{model}}}$, KPA represents the key, value, and query slices as normalized sums of contextual Kronecker products:

$$\mathbf{Q}_t = \frac{1}{R_Q} \sum_{r=1}^{R_Q} \mathbf{A}_r^Q(\mathbf{x}_t) \otimes \mathbf{B}_r^Q(\mathbf{x}_t) \in \mathbb{R}^{h \times d_h}, \quad (2)$$

$$\mathbf{K}_t = \frac{1}{R_K} \sum_{r=1}^{R_K} \mathbf{A}_r^K(\mathbf{x}_t) \otimes \mathbf{B}_r^K(\mathbf{x}_t) \in \mathbb{R}^{h \times d_h}, \quad (3)$$

$$\mathbf{V}_t = \frac{1}{R_V} \sum_{r=1}^{R_V} \mathbf{A}_r^V(\mathbf{x}_t) \otimes \mathbf{B}_r^V(\mathbf{x}_t) \in \mathbb{R}^{h \times d_h}, \quad (4)$$

where $\mathbf{A}_r(\mathbf{x}_t) \in \mathbb{R}^{h_1 \times d_1}$ operates on the head-partition subspace $\mathbb{R}^{h_1 \times d_1}$ and $\mathbf{B}_r(\mathbf{x}_t) \in \mathbb{R}^{h_2 \times d_2}$ operates on the within-head subspace $\mathbb{R}^{h_2 \times d_2}$. Both factors are fully contextual meaning they depend on $\mathbf{x}_t$ through independent learned linear projections as:

$$\mathbf{A}_r^K(\mathbf{x}_t) = \mathbf{W}_r^{A^K} \mathbf{x}_t, \quad \mathbf{B}_r^K(\mathbf{x}_t) = \mathbf{W}_r^{B^K} \mathbf{x}_t, \quad (5)$$

with $\mathbf{W}_r^{A^K} \in \mathbb{R}^{(h_1 d_1) \times d_{\text{model}}}$ and $\mathbf{W}_r^{B^K} \in \mathbb{R}^{(h_2 d_2) \times d_{\text{model}}}$ reshaped after multiplication. Analogous maps define the query and value factors. In practice, we merge the rank index into aggregate weight matrices $\mathbf{W}^{A^K} \in \mathbb{R}^{(R_K h_1 d_1) \times d_{\text{model}}}$ and $\mathbf{W}^{B^K} \in \mathbb{R}^{(R_K h_2 d_2) \times d_{\text{model}}}$, then reshape to $\mathcal{A}^K(\mathbf{x}_t) \in \mathbb{R}^{R_K \times h_1 \times d_1}$ and $\mathcal{B}^K(\mathbf{x}_t) \in \mathbb{R}^{R_K \times h_2 \times d_2}$, enabling a single batched projection per token.

The ranks R_Q, R_K, R_V are independent. Since queries are never cached, one may freely set $R_Q \gg R_K, R_V$ to enrich query expressivity at no memory cost. Only the $\mathbf{K}$ and $\mathbf{V}$ factors enter the cache.

It should be noted that setting $h_1 = h, d_1 = 1, h_2 = 1, d_2 = d_h$ makes $\mathbf{A}_r \in \mathbb{R}^{h \times 1}$ and $\mathbf{B}_r \in \mathbb{R}^{1 \times d_h}$ vectors, so each Kronecker term reduces to a rank-one outer product, exactly recovering TPA [Zhang et al., 2025]. Non-degenerate Kronecker factors are strictly richer: $\text{rank}(\mathbf{A}_r \otimes \mathbf{B}_r) = \text{rank}(\mathbf{A}_r) \cdot \text{rank}(\mathbf{B}_r)$, giving KPA strictly greater representational capacity at matched parameter budgets.

2.3 Factored Attention Score Computation

Heads in KPA are indexed by pairs (i_1, i_2) with $i_1 \in [h_1], i_2 \in [h_2]$. Denoting the i_1 -th row of $\mathbf{A}_r^Q$ as $(\mathbf{A}_r^Q)_{i_1} \in \mathbb{R}^{d_1}$, the mixed-product identity (1) gives an exact factorisation of the attention score (scaled by $1/\sqrt{d_h}$ before softmax in practice):

Proposition 2.2 (Factored Score Computation). *The attention score for head (i_1, i_2) at positions t and s satisfies*

$$\text{score}_{(i_1, i_2)}(t, s) = \frac{1}{R_Q R_K} \sum_{r=1}^{R_Q} \sum_{r'=1}^{R_K} \underbrace{(\mathbf{A}_r^Q)_{i_1}^\top (\mathbf{A}_{r'}^K)_{i_1}}_{\text{head-level}} \underbrace{(\mathbf{B}_r^Q)_{i_2}^\top (\mathbf{B}_{r'}^K)_{i_2}}_{\text{feature-level}}. \quad (6)$$

Proof. See Appendix A. □

Every attention score decomposes into a product of a *head-level* and a *feature-level* inner product, reflecting the two-level structure of KPA. Proposition 2.2 shows that scores can in principle be computed directly from the compact factors without materializing the full key tensor, analogously to factored score computation in TPA [Zhang et al., 2025].

2.4 Kronecker-Separable RoPE

A natural concern with any structured KV factorisation is compatibility with rotary positional embeddings (RoPE) [Su et al., 2024]. MLA resolves this by maintaining a small set of non-compressed KV dimensions that carry the rotary encoding, introducing a decoupled RoPE branch at every decoding step. We show that KPA requires no such branch.

Definition 2.3 (Kronecker-Separable RoPE). Let $\mathbf{I}_{d_1}$ be the $d_1 \times d_1$ identity and $\mathbf{R}_\theta^{(2)}(t) \in \mathbb{R}^{d_2 \times d_2}$ be a RoPE rotation matrix with frequencies calibrated to d_2 : $\theta_j = 10000^{-2j/d_2}$, $j = 0, \dots, d_2/2 - 1$. The Kronecker-Separable RoPE operator on $\mathbb{R}^{d_h}$ is $\mathbf{R}_{\text{KPA}}(t) = \mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(t)$.

Theorem 2.4 (RoPE Compatibility of KPA). Under Kronecker-Separable RoPE, applying $\mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(\cdot)$ to the query or key vector of head (i_1, i_2) is equivalent to rotating only the $\mathbf{B}$ factor:

$$(\mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(t))((\mathbf{A}_r^Q)_{i_1} \otimes (\mathbf{B}_r^Q)_{i_2}) = (\mathbf{A}_r^Q)_{i_1} \otimes \mathbf{R}_\theta^{(2)}(t)(\mathbf{B}_r^Q)_{i_2}, \quad (7)$$

$$(\mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(s))((\mathbf{A}_{r'}^K)_{i_1} \otimes (\mathbf{B}_{r'}^K)_{i_2}) = (\mathbf{A}_{r'}^K)_{i_1} \otimes \mathbf{R}_\theta^{(2)}(s)(\mathbf{B}_{r'}^K)_{i_2}. \quad (8)$$

Moreover, the attention score depends only on the relative offset $s - t$:

$$\text{score}_{(i_1, i_2)}(t, s) = \frac{1}{R_Q R_K} \sum_{r, r'} [(\mathbf{A}_r^Q)_{i_1}^\top (\mathbf{A}_{r'}^K)_{i_1}] \cdot [(\mathbf{B}_r^Q)_{i_2}^\top \mathbf{R}_\theta^{(2)}(s - t) (\mathbf{B}_{r'}^K)_{i_2}]. \quad (9)$$

Proof. See Appendix A. □

Theorem 2.4 has two practical consequences: (i) only the $\mathbf{B}^K$ factors need to be pre-rotated before caching; the $\mathbf{A}^K$ factors are position-independent and stored as-is, and (ii) the $\mathbf{B}$ factor structure is preserved by the rotation, so positional encoding need not require reconstructing the full key tensor. Unlike MLA, no decoupled RoPE branch is required.

2.5 KV Cache Cost and Compression

During autoregressive decoding, KPA caches the key and value factors per token t :

$$\{\mathbf{A}_r^K(\mathbf{x}_t), \mathbf{B}_r^K(\mathbf{x}_t)\}_{r=1}^{R_K} \cup \{\mathbf{A}_r^V(\mathbf{x}_t), \mathbf{B}_r^V(\mathbf{x}_t)\}_{r=1}^{R_V},$$

where $\mathbf{A}_r^K \in \mathbb{R}^{h_1 \times d_1}$ and $\mathbf{B}_r^K \in \mathbb{R}^{h_2 \times d_2}$ (pre-rotated via Kronecker-Separable RoPE). The per-token cache cost is $(R_K + R_V)(h_1 d_1 + h_2 d_2)$, compared with $2h d_h = 2h_1 h_2 d_1 d_2$ for MHA. The compression factor is:

$$C_{\text{KPA}} = \frac{2h_1 h_2 d_1 d_2}{(R_K + R_V)(h_1 d_1 + h_2 d_2)}. \quad (10)$$

For fixed h and d_h , compression increases with factor asymmetry: a more unequal split of h or d_h makes $h_1 d_1 + h_2 d_2$ smaller relative to $h_1 h_2 d_1 d_2$.

3 Experiments

3.1 Experimental Setup

We train KPA models at four scales (125M, 353M, 773M, 1.5B) using a Llama-inspired architecture with SwiGLU and RoPE on top of nanoGPT [Karpathy, 2022], trained on FineWeb-Edu for $\sim 50\text{B}$ tokens with $R_K = R_V = 2$ and $R_Q = 6$. We compare against MHA, MQA, GQA, MLA, and TPA baselines from Zhang et al. [2025] under identical conditions, evaluating zero-shot on nine benchmarks using `lm-evaluation-harness` [Gao et al., 2023]. Full training and evaluation details are in Appendix C.

3.2 Main Results

Tables 1 and 2 report zero-shot benchmark performance at 353M and 773M scales; 1.5B and 125M results appear in Appendix D.

At 353M, all three KPA configurations outperform TPA. KPA $h=(8 \times 2)$ achieves the best average at $6.4\times$ compression, matching TPA’s ratio while exceeding its accuracy by 1.49pp, with the largest gains on ARC-Easy and SciQ. KPA $h=(4 \times 4)$ outperforms TPA at strictly higher compression ($8\times$), demonstrating that increasing compression need not sacrifice accuracy. At 773M, KPA $h=(10 \times 2)$ outperforms TPA, MLA, and all other baselines at $6.7\times$ compression, with particularly strong gains on ARC-Easy, BoolQ, and SciQ. Notably, KPA surpasses MLA despite requiring no decoupled RoPE branch. KPA $h=(4 \times 5)$ achieves better accuracy than TPA at higher compression ($8.9\times$).

Table 1: Zero-shot evaluation on nine benchmarks (FineWeb-Edu, ~ 50 B tokens), 353M-parameter models. Best score per column in **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 $h=(4\times 4)$	64.23	30.72	59.66	44.09	34.40	70.08	54.38	26.79	85.30	52.18
KPA $h=(2\times 8)$	64.15	31.48	57.21	45.93	36.20	70.18	54.92	26.31	83.90	52.25
KPA $h=(8\times 2)$	66.16	33.02	60.37	45.11	35.80	71.29	53.12	26.79	84.40	52.90

Table 2: Zero-shot evaluation on nine benchmarks (FineWeb-Edu, ~ 50 B tokens), 773M-parameter models. Best score per column in **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 $h=(4\times 5)$	68.56	35.92	60.09	50.28	36.80	72.85	55.33	25.26	85.90	54.55
KPA $h=(2\times 10)$	68.31	34.98	55.32	50.91	38.00	71.06	55.64	24.66	85.30	53.80
KPA $h=(10\times 2)$	68.89	36.01	62.03	50.38	39.40	72.85	55.98	24.52	86.00	55.12

In both cases, the best-performing KPA variant uses an asymmetric split with larger h_1 , consistent with the ablation findings in Section 3.3. KPA improves monotonically across scales and matches or exceeds TPA at comparable or higher compression ratios at every scale.

3.3 Ablation on Head-Split Asymmetry

We ablate the effect of the Kronecker head split $h_1 \times h_2 = h$, holding $d_1 = d_2 = 8$ and $R_K = R_V = 2$ fixed. At 353M we compare symmetric $h=(4\times 4)$, asymmetric $h=(2\times 8)$, and reversed asymmetric $h=(8\times 2)$; at 773M we compare $h=(4\times 5)$, $h=(2\times 10)$, and $h=(10\times 2)$. Tables 1 and 2 reveal a consistent pattern across scales. Asymmetric splits with larger h_1 outperform both symmetric KPA and TPA at both scales. At 353M, $h=(8\times 2)$ achieves the best average, and at 773M, $h=(10\times 2)$ outperforms all baselines. We attribute this to the A -factor’s capacity to encode joint interactions across h_1 heads simultaneously in feature space. A larger h_1 expands this capacity, whereas the B -factor refines within-head feature structure independently, which appears less critical. This is distinct from MHA, where head projections are fully independent by construction. Symmetric splits trade some accuracy for higher compression ($8\times$ to $8.9\times$), offering a favourable operating point when memory is the primary constraint. The head split $h_1 \times h_2$ is a tuneable hyperparameter that allows practitioners to navigate the compression–accuracy trade-off at design time, with no other architectural changes required.

4 Conclusion

We introduced Kronecker Product Attention (KPA), a structured factorization of key and value projections that strictly generalizes TPA and integrates natively with RoPE without requiring a decoupled branch as in MLA. KPA matches or outperforms TPA, MHA, MQA, GQA, and MLA across all evaluated scales under identical training conditions, while achieving substantial KV cache compression. The Kronecker head split is a tuneable hyperparameter that allows practitioners to navigate the compression–accuracy trade-off at design time without architectural changes. By reducing the KV cache footprint, KPA directly addresses the memory-bandwidth bottleneck that limits language models deployment on resource-constrained devices such as smartphones, robotics platforms, and embedded systems, making it a natural fit for agentic systems that must plan and act under strict memory budgets without relying on cloud infrastructure. The same compression benefits apply during fine-tuning for tool use and multi-step reasoning, where longer context windows are critical but memory is scarce. Developing an efficient factored decoding kernel and evaluating KPA on longer contexts where cache compression is most practically impactful are natural next steps.

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*, pages 4895–4901, 2023.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. 2023.
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with Performers. In *International Conference on Learning Representations (ICLR)*, 2021.
- 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, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. KronA: Parameter efficient tuning with Kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022a.
- Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for GPT compression. In *ACL*, 2022b.
- 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.
- 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.
- Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- Alexander Novikov, Dmitry Podoprikin, 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.
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

- Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.
- 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.
- Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
- Marzieh Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kronecker-BERT: 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 (NAACL)*, pages 2116–2127, 2022.
- Charles F. Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear Algebra for Large Scale and Real-Time Applications*, pages 293–314. Springer, 1993.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.
- Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. 2020.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.
- 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.
- 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 Proofs

A.1 Proof of Proposition 2.2

Proof. For head (i_1, i_2) , the per-head query and key vectors are

$$\mathbf{q}_{t,(i_1,i_2)} = \frac{1}{R_Q} \sum_r (\mathbf{A}_r^Q)_{i_1} \otimes (\mathbf{B}_r^Q)_{i_2}, \quad \mathbf{k}_{s,(i_1,i_2)} = \frac{1}{R_K} \sum_{r'} (\mathbf{A}_{r'}^K)_{i_1} \otimes (\mathbf{B}_{r'}^K)_{i_2}.$$

Taking the dot product and applying the mixed-product identity

$$(\mathbf{a} \otimes \mathbf{b})^\top (\mathbf{c} \otimes \mathbf{d}) = (\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{d}),$$

we obtain

$$\mathbf{q}_{t,(i_1,i_2)}^\top \mathbf{k}_{s,(i_1,i_2)} = \frac{1}{R_Q R_K} \sum_{r,r'} [(\mathbf{A}_r^Q)_{i_1}^\top (\mathbf{A}_{r'}^K)_{i_1}] [(\mathbf{B}_r^Q)_{i_2}^\top (\mathbf{B}_{r'}^K)_{i_2}]. \quad \square$$

A.2 Proof of Theorem 2.4

Proof. Applying $\mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(t)$ to $(\mathbf{A}_r^Q)_{i_1} \otimes (\mathbf{B}_r^Q)_{i_2}$ and using the mixed-product identity gives

$$(\mathbf{I}_{d_1} \otimes \mathbf{R}_\theta^{(2)}(t))((\mathbf{A}_r^Q)_{i_1} \otimes (\mathbf{B}_r^Q)_{i_2}) = (\mathbf{A}_r^Q)_{i_1} \otimes \mathbf{R}_\theta^{(2)}(t)(\mathbf{B}_r^Q)_{i_2}.$$

Summing over r and averaging by $1/R_Q$ preserves the factored form.

The relative-position identity then follows from

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

which is the standard RoPE relative-position property restricted to the d_2 -dimensional subspace. $\square$

B KV Cache Compression

During autoregressive decoding, KPA caches the factor matrices per past token at a cost of $(R_K + R_V)(h_1 d_1 + h_2 d_2)$ per token, compared with $2h_1 h_2 d_1 d_2$ for MHA. The compression factor is

$$C_{\text{KPA}} = \frac{2h_1 h_2 d_1 d_2}{(R_K + R_V)(h_1 d_1 + h_2 d_2)}. \quad (11)$$

Table 3 reports C_{KPA} for all evaluated configurations with $R_K = R_V = 2$ and $d_1 = d_2 = 8$.

Table 3: KV cache compression factors for evaluated KPA configurations.

Scale	Config	C_{KPA}
353M	$h=(4 \times 4)$	$8.0 \times$
353M	$h=(2 \times 8)$	$6.4 \times$
353M	$h=(8 \times 2)$	$6.4 \times$
773M	$h=(4 \times 5)$	$8.9 \times$
773M	$h=(2 \times 10)$	$6.7 \times$
773M	$h=(10 \times 2)$	$6.7 \times$
1.5B	$h=(5 \times 5)$	$10.0 \times$
TPA	all scales	$6.4 \times$

C Training and Evaluation Details

Architecture and data. All models use the nanoGPT codebase [Karpathy, 2022] with SwiGLU activations [Shazeer, 2020] and RoPE embeddings [Su et al., 2024]. We fix $d_1 = d_2 = 8$ and vary h_1, h_2 across configurations to study head-split asymmetry. Models are trained from scratch on FineWeb-Edu-100B [Penedo et al., 2024] for 50,000 iterations with a batch size yielding approximately 1M tokens per iteration, totalling ~ 50 B tokens.

Optimisation. We use AdamW with peak learning rate 6×10^{-4} (3×10^{-4} at 773M and 1.5B scales), cosine decay to one-tenth of the peak, 2,000 warmup steps, and weight decay 0.1. All models train in bfloat16 on NVIDIA H100 GPUs.

Evaluation. We evaluate zero-shot performance on nine benchmarks using lm-evaluation-harness [Gao et al., 2023]: ARC-Easy, ARC-Challenge, BoolQ, HellaSwag, OpenbookQA, PIQA, WinoGrande, MMLU, and SciQ. We report accuracy-normalized scores for ARC-E, ARC-C, HellaSwag, OBQA, PIQA, and SciQ, and standard accuracy for BoolQ, WinoGrande, and MMLU. The primary scalar metric is the unweighted average across all nine tasks.

D Additional Experimental Results

D.1 1.5B Scale

Since $h=25$ admits only the symmetric Kronecker split $h_1=h_2=5$, a single KPA configuration is evaluated at the 1.5B scale. KPA outperforms TPA and all other baselines on eight of nine benchmarks, with the only exception being MMLU where MHA achieves a slightly higher score. The gains are most pronounced on ARC-Easy, SciQ, and BoolQ, consistent with the pattern observed at 353M and 773M scales. This confirms that KPA scales favourably with model size even when the head split is constrained to a single symmetric factorization.

D.2 125M Scale

At 125M scale, all KPA configurations outperform TPA (46.21%). Consistent with the pattern at larger scales, asymmetric splits with larger h_1 perform better: $h=(6 \times 2)$ achieves the best average, followed by $h=(4 \times 3)$, $h=(3 \times 4)$, and $h=(2 \times 6)$. The gains over TPA are smaller than at 353M and 773M, which is expected as smaller models have less capacity to benefit from richer factorization. Nevertheless, KPA consistently outperforms all baselines including MHA, MQA, GQA, MLA, and TPA across all four configurations.

Table 4: Zero-shot evaluation of 1.5B-parameter models (FineWeb-Edu, ~ 50 B tokens). Baselines from Zhang et al. [2025]. Best score in each column is **bold**.

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 $h=(5 \times 5)$	70.62	38.74	63.26	54.60	41.00	73.72	57.20	24.38	85.50	56.56

Table 5: Zero-shot evaluation of 125M-parameter models (FineWeb-Edu, ~ 50 B tokens). Baselines from Zhang et al. [2025]. Best score in each column is **bold**.

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	46.59
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	46.51
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

E Related Work

E.1 Head sharing

MQA [Shazeer, 2019] reduces the KV cache by sharing a single key and value head across all query heads, and GQA [Ainslie et al., 2023] generalizes this by grouping query heads to share a smaller set of key-value heads, recovering a spectrum between MQA and full MHA. MLA [DeepSeek-AI, 2024] instead projects keys and values into a low-dimensional latent space before caching, achieving stronger compression, but requires a decoupled RoPE branch to handle positional encoding. KPA avoids head sharing entirely and eliminates the decoupled RoPE branch by exploiting the Kronecker structure of its projections.

E.2 Low-rank and structured attention

Linformer [Wang et al., 2020] approximates the attention matrix via low-rank projection of keys and values along the sequence dimension, while Performer [Choromanski et al., 2021] approximates the softmax kernel with random features. These methods target the quadratic complexity of attention rather than KV cache size and do not apply to autoregressive decoding. Monarch [Dao et al., 2022] and Tensorizing-NN [Novikov et al., 2015] apply structured matrix factorizations to weight matrices for parameter efficiency, but do not specifically target KV cache compression. TPA [Zhang et al., 2025] is the closest prior work, applying CP decomposition to KV projections; KPA strictly generalizes TPA by replacing rank-1 vector outer products with Kronecker matrix factors.

E.3 KV compression, eviction, and quantization

H2O [Zhang et al., 2023] and StreamingLLM [Xiao et al., 2024] reduce cache size by evicting tokens deemed less important based on attention scores, but risk discarding tokens that become relevant later. KIVI [Liu et al., 2024] and KVQuant [Hooper et al., 2024] compress the cache through low-bit quantization, trading precision for memory. PagedAttention [Kwon et al., 2023] improves memory management through virtual paging but does not reduce the total cache footprint. KPA takes a complementary architectural approach, reducing the cache at the representation level before any eviction or quantization is applied.

E.4 Positional encoding

RoPE [Su et al., 2024] encodes relative position by rotating query and key vectors and has become the dominant positional scheme in modern language models. ALiBi [Press et al., 2022] instead adds position-dependent biases to attention logits, avoiding explicit positional embeddings. Extensions such as YaRN [Peng et al., 2024], xPos [Sun et al., 2023], and PI [Chen et al., 2023] adapt RoPE for longer contexts through frequency scaling or interpolation. KPA introduces Kronecker-Separable RoPE, which applies rotary embeddings directly to the factorized representation by rotating only the feature-level factor, preserving the standard relative-position property without materializing full key tensors or requiring a decoupled positional branch.

E.5 Kronecker factorization in transformers.

Kronecker decomposition has been applied to transformers in several distinct settings, all targeting static weight compression rather than inference-time KV cache reduction. KroneckerBERT [Tahaei et al., 2022] compresses BERT weight matrices via Kronecker decomposition with knowledge distillation. KnGPT2 [Edalati et al., 2022b] applies the same idea to GPT-2, replacing linear layer weights with Kronecker-structured matrices and using intermediate layer distillation to recover accuracy. KronA [Edalati et al., 2022a] uses Kronecker products as a drop-in replacement for LoRA adapters in parameter-efficient fine-tuning, achieving a better expressivity-to-parameter trade-off than low-rank decomposition. HOT [Omranpour et al., 2024] decomposes the attention score matrix as a sum of Kronecker products to reduce training-time computation for multi-dimensional encoder inputs. KPA differs from all of these: it factorizes dynamic, per-token key and value activations rather than static weight matrices or score matrices, targeting inference-time memory reduction in autoregressive decoding, a setting none of the above address.