MULTI-HEAD LOW-RANK ATTENTION

Songtao Liu[1]* **Hongwu Peng**[2] **Zhiwei Zhang**[1] **Zhengyu Chen**[3] **Yue Guo**[4]
[1]The Pennsylvania State University [2]University of Connecticut
[3]Carnegie Mellon University [4]University of California, Los Angeles

ABSTRACT

Long-context inference in large language models is bottlenecked by Key–Value (KV) cache loading during the decoding stage, where the sequential nature of generation requires repeatedly transferring the KV cache from off-chip High-Bandwidth Memory (HBM) to on-chip Static Random-Access Memory (SRAM) at each step. While Multi-Head Latent Attention (MLA) significantly reduces the total KV cache size, it suffers from a sharding bottleneck during distributed decoding via Tensor Parallelism (TP). Since its single latent head cannot be partitioned, each device is forced to redundantly load the complete KV cache for every token, consuming excessive memory traffic and diminishing TP benefits like weight sharding. In this work, we propose Multi-Head Low-Rank Attention (MLRA), which enables partitionable latent states for efficient 4-way TP decoding. Extensive experiments show that MLRA achieves state-of-the-art perplexity and downstream task performance, while also delivering a 2.8× decoding speedup over MLA. Code is available at https://github.com/SongtaoLiu0823/MLRA. Pretrained weights, along with the training and evaluation data, are available at https://huggingface.co/Soughing/MLRA.

1 INTRODUCTION

Inference-time scaling (OpenAI et al., 2024) is critical for large language models (LLMs) to produce high-quality responses. Both retrieval-augmented generation (RAG) (Lewis et al., 2020) and long chain-of-thought (CoT) reasoning (Wei et al., 2022) rely on maintaining long context before generating the final answer, substantially increasing the number of tokens that must be processed at each decoding step. Sequential token generation under Multi-Head Attention (MHA) (Vaswani et al., 2017) requires reloading the Key–Value (KV) cache from high-bandwidth memory every step, so data movement (Ivanov et al., 2021; Gholami et al., 2024), not computation, dominates latency for long-context inference (Sadhukhan et al., 2025). The small amount of compute per step relative to this data movement leads to poor GPU utilization (He & Zhai, 2024; Zadouri et al., 2025).

A series of recent studies (Shazeer, 2019; Hu et al., 2024; Zadouri et al., 2025; Zhang et al., 2025; Zheng et al., 2025) have developed alternative attention mechanisms aimed at improving decoding efficiency and overall model quality. Multi-Head Latent Attention (MLA) (DeepSeek et al., 2024c) compresses the KV cache into a latent head ($4.5d_h$ per token). By absorbing the up-projection matrices into the queries during decoding, it delivers better efficiency compared with MHA. However, MLA is unfriendly to tensor parallelism (TP) because its single latent head cannot be sharded. In this work, we address the limitation that MLA does not support TP.

We first show that partitioning the MLA latent head and the NoPE (Yang et al., 2025b) KV up-projection matrices into four blocks makes the NoPE key and value equivalent to the sum of four block-wise projections. Motivated by this insight, we propose Multi-Head Low-Rank Attention (MLRA), which explicitly decomposes the latent head into four latent heads, independently up-projects each latent head to form NoPE KV, and sums the resulting attention outputs. This design naturally supports 4-way TP and reduces the per-device KV cache loading. Based on our 2.9B scale experiments, MLRA-4 achieves the lowest perplexity (13.672 vs. 13.727 for MLA and 14.139 for GQA) and highest zero-shot common-sense reasoning accuracy (58.84% vs. 58.75% for MLA and 57.89% for GQA). Our kernel delivers a 1.05-1.26× speedup over GQA in long-context decoding.

*Correspondence to: Songtao Liu <skl5761@psu.edu>.

2 BACKGROUND

2.1 MULTI-HEAD LATENT ATTENTION

All notation used in this paper is summarized in Appendix A. Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, MLA derives the query and KV states as follows:

$$\boldsymbol{C}^{\mathrm{Q}} = \mathrm{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{DQ}}\right), \quad \boldsymbol{W}^{\mathrm{DQ}} \in \mathbb{R}^{d\times d_c'},$$
$$\boldsymbol{C}^{\mathrm{KV}} = \mathrm{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{DKV}}\right), \quad \boldsymbol{W}^{\mathrm{DKV}} \in \mathbb{R}^{d\times d_c},$$
$$\boldsymbol{K}^{\mathrm{RoPE}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{KR}}\right), \quad \boldsymbol{W}^{\mathrm{KR}} \in \mathbb{R}^{d\times d_h^R},$$

where $d_c, d_c' \ll hd_h$ denote the dimensions for KV and query latent states, respectively. The learnable down-projection matrices $\boldsymbol{W}^{\mathrm{DQ}}$ and $\boldsymbol{W}^{\mathrm{DKV}}$ produce the latent states $\boldsymbol{C}^{\mathrm{Q}}$ and $\boldsymbol{C}^{\mathrm{KV}}$, while $\boldsymbol{W}^{\mathrm{KR}}$ generates the partial RoPE (Su et al., 2024) key, denoted as $\boldsymbol{K}^{\mathrm{RoPE}}$. Following DeepSeek-V3 (DeepSeek et al., 2024b), we set $d_c = 4d_h$ and $d_h^R = 0.5d_h$ without loss of generality. Both $\boldsymbol{C}^{\mathrm{KV}}$ and $\boldsymbol{K}^{\mathrm{RoPE}}$ are cached during inference to optimize efficiency. Finally, MLA derives the h attention heads for queries, NoPE keys, and values through the following up-projections:

$$\overline{\boldsymbol{Q}}^{\mathrm{NoPE}} = \boldsymbol{C}^{\mathrm{Q}}\boldsymbol{W}^{\mathrm{UQ}}, \quad \overline{\boldsymbol{Q}}^{\mathrm{RoPE}} = \mathrm{RoPE}\left(\boldsymbol{C}^{\mathrm{Q}}\boldsymbol{W}^{\mathrm{QR}}\right), \quad \boldsymbol{W}^{\mathrm{UQ}} \in \mathbb{R}^{d_c'\times(hd_h)}, \quad \boldsymbol{W}^{\mathrm{QR}} \in \mathbb{R}^{d_c'\times\left(hd_h^R\right)}$$
$$\overline{\boldsymbol{K}}^{\mathrm{NoPE}} = \boldsymbol{C}^{\mathrm{KV}}\boldsymbol{W}^{\mathrm{UK}}, \quad \overline{\boldsymbol{V}} = \boldsymbol{C}^{\mathrm{KV}}\boldsymbol{W}^{\mathrm{UV}}, \quad \boldsymbol{W}^{\mathrm{UK}}, \boldsymbol{W}^{\mathrm{UV}} \in \mathbb{R}^{d_c\times(hd_h)},$$

where $\boldsymbol{W}^{\mathrm{UQ}}$, $\boldsymbol{W}^{\mathrm{UK}}$, and $\boldsymbol{W}^{\mathrm{UV}}$ denote the learnable up-projection matrices. To facilitate multi-head computation, the resulting queries, NoPE keys, and values are reshaped into head-wise tensors:

$$\mathbf{Q}^{\mathrm{NoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{NoPE}}, [n, h, d_h]\right), \quad \mathbf{Q}^{\mathrm{RoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{RoPE}}, [n, h, d_h^R]\right)$$
$$\mathbf{K}^{\mathrm{NoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{K}}^{\mathrm{NoPE}}, [n, h, d_h]\right), \quad \mathbf{V} = \mathrm{Reshape}\left(\overline{\boldsymbol{V}}, [n, h, d_h]\right),$$

where $\mathbf{Q}^{\mathrm{NoPE}}, \mathbf{K}^{\mathrm{NoPE}}, \mathbf{V} \in \mathbb{R}^{n\times h\times d_h}$ and $\mathbf{Q}^{\mathrm{RoPE}} \in \mathbb{R}^{n\times h\times d_h^R}$. For head $i \in \{0, \ldots, h-1\}$, we define the head-specific 2D slices by indexing into the respective 3D tensors:

$$\mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} := \mathbf{Q}^{\mathrm{NoPE}}[:, i, :], \ \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} := \mathbf{Q}^{\mathrm{RoPE}}[:, i, :], \ \mathbf{K}^{\mathrm{NoPE}}_{:,i,:} := \mathbf{K}^{\mathrm{NoPE}}[:, i, :], \ \mathbf{V}_{:,i,:} := \mathbf{V}[:, i, :].$$

To incorporate positional information, MLA shares a common RoPE key $\boldsymbol{K}^{\mathrm{RoPE}}$ across all attention heads. The final position-aware query and key for head i are formed by concatenating their respective NoPE and RoPE components:

$$\mathbf{Q}_{:,i,:} = \mathrm{Concat}\left(\left[\mathbf{Q}^{\mathrm{NoPE}}_{:,i,:}, \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:}\right], \text{dim=1}\right), \quad \mathbf{K}_{:,i,:} = \mathrm{Concat}\left(\left[\mathbf{K}^{\mathrm{NoPE}}_{:,i,:}, \boldsymbol{K}^{\mathrm{RoPE}}\right], \text{dim=1}\right).$$

Efficient Decoding. Recall that MLA utilizes up-projection matrices $\boldsymbol{W}^{\mathrm{UK}}$ and $\boldsymbol{W}^{\mathrm{UV}}$. We extract the d_h-column slices for head i to define its head-specific projections:

$$\boldsymbol{W}^{\mathrm{UK}}_{:,(i)} := \boldsymbol{W}^{\mathrm{UK}}[:, id_h : (i+1)d_h], \quad \boldsymbol{W}^{\mathrm{UV}}_{:,(i)} := \boldsymbol{W}^{\mathrm{UV}}[:, id_h : (i+1)d_h].$$

We refer to the DeepSeek official inference implementation (DeepSeek et al., 2024b) to illustrate how to "absorb" up-projection matrices into the queries to avoid explicit KV materialization in MLA decoding. For the prefix sequence $\{0, \ldots, n-1\}$ with cached components $\boldsymbol{C}^{\mathrm{KV}}$ and $\boldsymbol{K}^{\mathrm{RoPE}}$, we define the head-wise up-projections by partitioning $\boldsymbol{W}^{\mathrm{UK}}$ and $\boldsymbol{W}^{\mathrm{UV}}$ into h heads, $\{\boldsymbol{W}^{\mathrm{UK}}_{:,(i)}\}_{i=0}^{h-1}$ and $\{\boldsymbol{W}^{\mathrm{UV}}_{:,(i)}\}_{i=0}^{h-1}$. For the last prefix token at position $n-1$, let $\mathbf{Q}_{n-1,i,:}$ denote the query vector for the i-th attention head. To maintain variance during the dot-product operation, we apply the scaling factor $\tau = \frac{1}{\sqrt{d_h + d_h^R}}$ and compute the attention output for the token at position $n-1$ as follows:

$$\mathbf{O}_{n-1,i,:} = \mathrm{Softmax}\left(\tau\mathbf{Q}^{\mathrm{NoPE}}_{n-1,i,:}\left(\boldsymbol{C}^{\mathrm{KV}}\boldsymbol{W}^{\mathrm{UK}}_{:,(i)}\right)^{\top} + \tau\mathbf{Q}^{\mathrm{RoPE}}_{n-1,i,:}\left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right)\left(\boldsymbol{C}^{\mathrm{KV}}\boldsymbol{W}^{\mathrm{UV}}_{:,(i)}\right),$$
$$= \mathrm{Softmax}\left(\tau\underbrace{\mathbf{Q}^{\mathrm{NoPE}}_{n-1,i,:}\left(\boldsymbol{W}^{\mathrm{UK}}_{:,(i)}\right)^{\top}}_{\tilde{\mathbf{Q}}^{\mathrm{NoPE}}_{n-1,i,:}\in\mathbb{R}^{d_c}}\left(\boldsymbol{C}^{\mathrm{KV}}\right)^{\top} + \tau\mathbf{Q}^{\mathrm{RoPE}}_{n-1,i,:}\left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right)\boldsymbol{C}^{\mathrm{KV}}\boldsymbol{W}^{\mathrm{UV}}_{:,(i)},$$

where $\mathbf{O}_{n-1,i,:}$ is the attention output for head i at position $n-1$. We next present a three-step algorithm that leverages the associativity of matrix multiplication to avoid materializing the h heads of NoPE keys and values, thereby optimizing the decoding efficiency.

Step 1 (Query-Side Weight Absorption). We first reorganize the NoPE key and value up-projection matrices into head-wise tensors:

$$\tilde{\mathbf{W}}^{\mathrm{UK}}_{i,:,:} := \left(\boldsymbol{W}^{\mathrm{UK}}_{:,(i)}\right)^{\top} \in \mathbb{R}^{d_h \times d_c}, \qquad \tilde{\mathbf{W}}^{\mathrm{UV}}_{i,:,:} := \boldsymbol{W}^{\mathrm{UV}}_{:,(i)} \in \mathbb{R}^{d_c \times d_h},$$

where $\tilde{\mathbf{W}}^{\mathrm{UK}} \in \mathbb{R}^{h \times d_h \times d_c}$ and $\tilde{\mathbf{W}}^{\mathrm{UV}} \in \mathbb{R}^{h \times d_c \times d_h}$. For the NoPE query at position $n-1$, $\mathbf{Q}^{\mathrm{NoPE}}_{n-1,:,:}$, we absorb the up-projection weight tensor $\tilde{\mathbf{W}}^{\mathrm{UK}}$ directly into the query via Einstein summation:

$$\tilde{\mathbf{Q}}^{\mathrm{NoPE}}_{n-1,:,:} = \mathrm{einsum}\left(\texttt{"hp,hpc->hc"}, \mathbf{Q}^{\mathrm{NoPE}}_{n-1,:,:}, \tilde{\mathbf{W}}^{\mathrm{UK}}\right), \quad p = d_h,\, c = d_c, \quad \tilde{\mathbf{Q}}^{\mathrm{NoPE}}_{n-1,:,:} \in \mathbb{R}^{h \times d_c}.$$

Step 2 (MQA-Style Decoding on Latent KV Cache). Given the KV cache $\boldsymbol{C}^{\mathrm{KV}}$ and $\boldsymbol{K}^{\mathrm{RoPE}}$, we define the shared key and value tensors by concatenating and reshaping the latent representations as $\tilde{\mathbf{K}} = \mathrm{Reshape}\left(\mathrm{Concat}\left(\left[\boldsymbol{C}^{\mathrm{KV}}, \boldsymbol{K}^{\mathrm{RoPE}}\right], \mathrm{dim{=}1}\right), \left[n, 1, d_c + d_h^R\right]\right) \in \mathbb{R}^{n \times 1 \times (d_c + d_h^R)}$ and $\tilde{\mathbf{V}} = \mathrm{Reshape}\left(\boldsymbol{C}^{\mathrm{KV}}, [n, 1, d_c]\right) \in \mathbb{R}^{n \times 1 \times d_c}$. Under this formulation, decoding reduces to an MQA-style attention mechanism in which the attention logits (i.e., query–key inner products before softmax) are computed in a $(d_c + d_h^R)$-dimensional space using these shared KV states. Incorporating the concatenated query $\tilde{\mathbf{Q}}_{n-1,:,:} = \mathrm{Concat}\left(\left[\tilde{\mathbf{Q}}^{\mathrm{NoPE}}_{n-1,:,:}, \mathbf{Q}^{\mathrm{RoPE}}_{n-1,:,:}\right], \mathrm{dim{=}1}\right) \in \mathbb{R}^{h \times (d_c + d_h^R)}$, the attention output is calculated as follows:

$$\mathbf{Z}_{n-1,:,:} = \mathrm{Attention}\left(\tilde{\mathbf{Q}}_{n-1,:,:}, \mathrm{RepeatInterleave}\left(\tilde{\mathbf{K}}, h, \mathrm{dim} = 1\right), \mathrm{RepeatInterleave}\left(\tilde{\mathbf{V}}, h, \mathrm{dim} = 1\right)\right), \tag{1}$$

where $\mathbf{Z}_{n-1,:,:} \in \mathbb{R}^{h \times d_c}$. FlashAttention-3 (Shah et al., 2024) and FlashMLA (Jiashi Li, 2025) provide highly optimized kernels designed to implement the Step-2 decoding computation directly.

Step 3 (Output Up-Projection). Finally, the up-projection tensor maps the intermediate attention output to the final attention output:

$$\mathbf{O}_{n-1,:,:} = \mathrm{einsum}\left(\texttt{"hc,hcp->hp"}, \mathbf{Z}_{n-1,:,:}, \tilde{\mathbf{W}}^{\mathrm{UV}}\right), \quad c = d_c,\, p = d_h, \quad \mathbf{O}_{n-1,:,:} \in \mathbb{R}^{h \times d_h}.$$

Block Multiplications. For each head i, we define the constituent sub-blocks $\boldsymbol{W}^{\cdot}_{(b),(i)} \in \mathbb{R}^{d_h \times d_h}$ by partitioning the up-projection matrices into d_h-sized row blocks for $b \in \{0, 1, 2, 3\}$:

$$\boldsymbol{W}^{\mathrm{UK}}_{(b),(i)} := \boldsymbol{W}^{\mathrm{UK}}\left[bd_h : (b+1)d_h,\, id_h : (i+1)d_h\right],$$
$$\boldsymbol{W}^{\mathrm{UV}}_{(b),(i)} := \boldsymbol{W}^{\mathrm{UV}}\left[bd_h : (b+1)d_h,\, id_h : (i+1)d_h\right].$$

Consequently, each head-specific up-projection matrix can be expressed as a vertical stack of these four row-blocks:

$$\boldsymbol{W}^{\mathrm{UK}}_{:,(i)} = \begin{bmatrix} \boldsymbol{W}^{\mathrm{UK}}_{(0),(i)} \\ \boldsymbol{W}^{\mathrm{UK}}_{(1),(i)} \\ \boldsymbol{W}^{\mathrm{UK}}_{(2),(i)} \\ \boldsymbol{W}^{\mathrm{UK}}_{(3),(i)} \end{bmatrix}, \quad \boldsymbol{W}^{\mathrm{UV}}_{:,(i)} = \begin{bmatrix} \boldsymbol{W}^{\mathrm{UV}}_{(0),(i)} \\ \boldsymbol{W}^{\mathrm{UV}}_{(1),(i)} \\ \boldsymbol{W}^{\mathrm{UV}}_{(2),(i)} \\ \boldsymbol{W}^{\mathrm{UV}}_{(3),(i)} \end{bmatrix}.$$

Similarly, we partition the KV latent matrix $\boldsymbol{C}^{\mathrm{KV}} \in \mathbb{R}^{n \times d_c}$ into horizontal channel blocks $\boldsymbol{C}^{\mathrm{KV}}_{:,(b)} := \boldsymbol{C}^{\mathrm{KV}}\left[:, bd_h : (b+1)d_h\right]$, such that $\boldsymbol{C}^{\mathrm{KV}} = \left[\boldsymbol{C}^{\mathrm{KV}}_{:,(0)}, \ldots, \boldsymbol{C}^{\mathrm{KV}}_{:,(3)}\right]$. This block decomposition allows the key and value projections for head i to be reformulated as a sum of four sub-block products:

$$\mathbf{K}^{\mathrm{NoPE}}_{:,(i),:} = \sum_{b=0}^{3} \boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UK}}_{(b),(i)}, \qquad \mathbf{V}_{:,(i),:} = \sum_{b=0}^{3} \boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UV}}_{(b),(i)}. \tag{2}$$

2.2 Grouped Latent Attention

Grouped Latent Attention (GLA-2) (Zadouri et al., 2025) bisects MLA's single latent head into two latent heads, using the first latent head $(\boldsymbol{C}^{\mathrm{KV}}_{:,(0)}, \boldsymbol{C}^{\mathrm{KV}}_{:,(1)})$ for the first half of attention heads and the second latent head $(\boldsymbol{C}^{\mathrm{KV}}_{:,(2)}, \boldsymbol{C}^{\mathrm{KV}}_{:,(3)})$ for the second half. We define the group-mapping function as:

$$\gamma(i) = \begin{cases} 0, & i < h/2, \\ 1, & i \geq h/2, \end{cases} \qquad \bar{i} = i - \frac{\gamma(i)\, h}{2} \in \{0, \ldots, h/2 - 1\}. \tag{3}$$

Let $\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}, \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}} \in \mathbb{R}^{2d_h \times (h/2)\, d_h}$ denote the up-projection matrices for latent group $\gamma(i) \in \{0, 1\}$. We extract the head-specific slices for head i by indexing into these matrices:

$$\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{:,(i)} = \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}\left[:,\, \bar{i}d_h : (\bar{i}+1)d_h\right], \quad \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{:,(i)} = \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}\left[:,\, \bar{i}d_h : (\bar{i}+1)d_h\right],$$

where $\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{:,(i)}, \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{:,(i)} \in \mathbb{R}^{2d_h \times d_h}$. To further facilitate block-wise computation, we partition these slices into d_h-row blocks $\boldsymbol{W}^{(\gamma(i)),\cdot}_{(b),(i)}$ for $b \in \{0, 1\}$, defined as:

$$\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(b),(i)} := \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}\left[bd_h : (b+1)d_h,\, \bar{i}d_h : (\bar{i}+1)d_h\right],$$
$$\boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(b),(i)} := \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}\left[bd_h : (b+1)d_h,\, \bar{i}d_h : (\bar{i}+1)d_h\right],$$

where each block $\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(b),(i)}, \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(b),(i)} \in \mathbb{R}^{d_h \times d_h}$. This partitioning allows us to decompose the head-specific up-projection matrices into two row-wise blocks:

$$\boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{:,(i)} = \begin{bmatrix} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(0),(i)} \\ \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(1),(i)} \end{bmatrix}, \qquad \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{:,(i)} = \begin{bmatrix} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(0),(i)} \\ \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(1),(i)} \end{bmatrix}.$$

Consequently, the NoPE key and value computations for head i can be expressed as the summation of two block products:

$$\begin{aligned} \mathbf{K}^{\mathrm{NoPE}}_{:,(i),:} &= \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(0),(i)} + \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(1),(i)}, \\ \mathbf{V}_{:,(i),:} &= \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(0),(i)} + \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(1),(i)}. \end{aligned} \tag{4}$$

3 Multi-Head Low-Rank Attention

Building on the block decompositions in Sections 2.1 and 2.2, we propose MLRA. By shifting the summation from KV computation to attention output, MLRA treats each block projection as an independent low-rank branch and sums their outputs. MLRA is illustrated in Figures 8 and 9.

3.1 MLRA-4

By substituting the block-partitioned identities from Eq. (2) into the attention mechanism, the output for head i can be expressed as:

$$\mathbf{O}_{:,i,:} = \mathrm{Softmax}\left(\tau \mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} \left(\sum_{b=0}^{3} \boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UK}}_{(b),(i)}\right)^{\top} + \tau \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} \left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right) \left(\sum_{b=0}^{3} \boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UV}}_{(b),(i)}\right).$$

Motivated by Eq. (2), we propose MLRA-4, which computes attention independently on each block-wise branch and sums the resulting outputs:

$$\mathbf{O}_{:,i,:} = \sum_{b=0}^{3} \mathrm{Softmax}\left(\tau \mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UK}}_{(b),(i)}\right)^{\top} + \tau \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} \left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right) \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(b)} \boldsymbol{W}^{\mathrm{UV}}_{(b),(i)}\right). \tag{5}$$

3.2 MLRA-2

Following the grouping logic of GLA-2 from Eq. (3) and substituting the block-wise identities from Eq. (4), the attention output for GLA-2 can be expressed as:

$$\begin{aligned} \mathbf{O}_{:,i,:} = \mathrm{Softmax}&\left(\tau \mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(0),(i)} + \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(1),(i)}\right)^{\top} + \tau \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} \left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right) \\ &\cdot \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(0),(i)} + \boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(1),(i)}\right). \end{aligned}$$

Analogously, we derive MLRA-2 by moving the block summation outside the attention operator, yielding a sum of two branchwise attention outputs:

$$\begin{aligned} \mathbf{O}_{:,i,:} = \mathrm{Softmax}&\left(\tau \mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(0),(i)}\right)^{\top} + \tau \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} \left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right) \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i))} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(0),(i)}\right) \\ + \mathrm{Softmax}&\left(\tau \mathbf{Q}^{\mathrm{NoPE}}_{:,i,:} \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UK}}_{(1),(i)}\right)^{\top} + \tau \mathbf{Q}^{\mathrm{RoPE}}_{:,i,:} \left(\boldsymbol{K}^{\mathrm{RoPE}}\right)^{\top}\right) \left(\boldsymbol{C}^{\mathrm{KV}}_{:,(2\gamma(i)+1)} \boldsymbol{W}^{(\gamma(i)),\mathrm{UV}}_{(1),(i)}\right). \end{aligned} \tag{6}$$

MLRA-2 and MLRA-4 differ primarily in their latent-to-head mapping and branching factor. In MLRA-2, each latent block is up-projected to serve $h/2$ heads, with the final output resulting from a two-branch summation. Conversely, MLRA-4 utilizes four latent blocks that are each up-projected to serve all h heads, resulting in a four-branch summation. Despite these structural differences, both variants decompose the computation into independent branches that require only a final reduction. This architecture naturally facilitates 4-way TP decoding, reducing the per-head attention logit space to $1.5d_h$ after absorption—a significant reduction compared to $4.5d_h$ in MLA and $2.5d_h$ in GLA-2.

3.3 Scaling Query/Key–Value Latent States and Attention Output

Recent work (LongCat et al., 2025) observes that the RoPE key $\left(\boldsymbol{K}^{\text{RoPE}}\right)$ can exhibit a significant variance mismatch relative to other attention components $\left(\mathbf{Q}, \mathbf{K}^{\text{NoPE}}, \mathbf{V}\right)$. This discrepancy arises in MLA because RMSNorm (Zhang & Sennrich, 2019) is applied to the latent states $\boldsymbol{H}\boldsymbol{W}^{\text{DQ}}$ and $\boldsymbol{H}\boldsymbol{W}^{\text{DKV}}$ prior to the up-projections that generate the final query, NoPE key, and value tensors. To formally investigate this variance instability, we introduce the following assumption:

Assumption 1. *We assume that all elements of the weight matrices $\boldsymbol{W}^{DQ}$, $\boldsymbol{W}^{DKV}$, $\boldsymbol{W}^{UQ}$, $\boldsymbol{W}^{QR}$, $\boldsymbol{W}^{UK}$, $\boldsymbol{W}^{KR}$, and $\boldsymbol{W}^{UV}$ are independent and identically distributed (i.i.d.) random variables with zero mean and variance σ_w^2. Furthermore, these weight matrices are assumed to be mutually independent of the input signals at each layer. Finally, for the multi-branch of MLRA, we assume the attention outputs $\mathbf{O}_{(b),(i),:}$ originating from different latent blocks b are mutually uncorrelated, implying that* $\operatorname{Cov}\left(\mathbf{O}_{(a),(i),:}, \mathbf{O}_{(b),(i),:}\right) \approx 0$ *for all* $a \neq b$.

RoPE Key Variance. Recall that MLA computes the RoPE key as $\boldsymbol{K}^{\text{RoPE}} = \operatorname{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\text{KR}}\right)$. Let $\overline{\boldsymbol{K}}^{\text{RoPE}}_{t,u} := (\boldsymbol{H}\boldsymbol{W}^{\text{KR}})_{t,u} = \sum_{m=1}^{d} \boldsymbol{H}_{t,m}\boldsymbol{W}^{\text{KR}}_{m,u}$. Since the hidden states $\boldsymbol{H}$ are RMS-normalized, $\mathbb{E}\left[(\boldsymbol{H}_{t,m})^2\right] \approx 1$. With $\mathbb{E}\left[\boldsymbol{W}^{\text{KR}}_{m,u}\right] = 0$ and $\mathbb{E}\left[\left(\boldsymbol{W}^{\text{KR}}_{m,u}\right)^2\right] = \sigma_w^2$, we have

$$\operatorname{Var}\left(\overline{\boldsymbol{K}}^{\text{RoPE}}_{t,u}\right) = \sum_{m=1}^{d}\left(\mathbb{E}\left[(\boldsymbol{H}_{t,m})^2\right]\,\mathbb{E}\left[(\boldsymbol{W}^{\text{KR}}_{m,u})^2\right] - \left(\mathbb{E}[\boldsymbol{H}_{t,m}]\,\mathbb{E}[\boldsymbol{W}^{\text{KR}}_{m,u}]\right)^2\right) \approx \sum_{m=1}^{d} 1 \cdot \sigma_w^2 = d\sigma_w^2.$$

Since RoPE is an orthogonal transformation, it does not change the variance:

$$\operatorname{Var}\left(\boldsymbol{K}^{\text{RoPE}}\right) \approx d\sigma_w^2. \tag{7}$$

NoPE Key Variance. Next, we derive the variance of the NoPE keys, $\overline{\boldsymbol{K}}^{\text{NoPE}} = \boldsymbol{C}^{\text{KV}}\boldsymbol{W}^{\text{UK}}$. By definition, the latent KV state $\boldsymbol{C}^{\text{KV}} = \operatorname{RMSNorm}(\boldsymbol{H}\boldsymbol{W}^{\text{DKV}})$ is constrained such that each element has approximately unit mean square, i.e., $\mathbb{E}\left[(\boldsymbol{C}^{\text{KV}}_{t,l})^2\right] \approx 1$. Considering an arbitrary entry $\overline{\boldsymbol{K}}^{\text{NoPE}}_{t,u} = \sum_{l=1}^{d_c} \boldsymbol{C}^{\text{KV}}_{t,l}\boldsymbol{W}^{\text{UK}}_{l,u}$, and assuming the weights $\boldsymbol{W}^{\text{UK}}$ are zero-mean with variance σ_w^2, the variance of the product is:

$$\operatorname{Var}\left(\overline{\boldsymbol{K}}^{\text{NoPE}}_{t,u}\right) = \sum_{l=1}^{d_c}\left(\mathbb{E}\left[(\boldsymbol{C}^{\text{KV}}_{t,l})^2\right]\,\mathbb{E}\left[(\boldsymbol{W}^{\text{UK}}_{l,u})^2\right] - \left(\mathbb{E}[\boldsymbol{C}^{\text{KV}}_{t,l}]\,\mathbb{E}[\boldsymbol{W}^{\text{UK}}_{l,u}]\right)^2\right) \approx \sum_{l=1}^{d_c} 1 \cdot \sigma_w^2 = d_c\sigma_w^2. \tag{8}$$

Because reshaping does not alter the underlying variance, $\operatorname{Var}\left(\mathbf{K}^{\text{NoPE}}\right)$ remains $d_c\sigma_w^2$. Extending this derivation to the remaining attention components, we obtain the variances for the value and query tensors as follows:

$$\operatorname{Var}(\mathbf{V}) \approx d_c\sigma_w^2, \qquad \operatorname{Var}\left(\mathbf{Q}^{\text{NoPE}}\right) \approx d'_c\sigma_w^2, \qquad \operatorname{Var}\left(\mathbf{Q}^{\text{RoPE}}\right) \approx d'_c\sigma_w^2.$$

Variance Mismatch and Calibration. Comparing our variance derivations shows that $\frac{\operatorname{Var}\left(\boldsymbol{K}^{\text{RoPE}}\right)}{\operatorname{Var}\left(\mathbf{K}^{\text{NoPE}}\right)} \approx \frac{d}{d_c}$, which explains the mismatch noted in LongCat et al. (2025) when the latent dimension d_c is much smaller than d. This is corrected by applying scaling factors to the latent states before the up-projection. Specifically, using $\alpha_q = \sqrt{d/d'_c}$ and $\alpha_{kv} = \sqrt{d/d_c}$ ensures that the query and NoPE key components $\left(\mathbf{Q}, \mathbf{K}^{\text{NoPE}}\right)$ achieve parity with the variance of the RoPE key.

Table 1: Comparison of parameters and KV cache loading among attention mechanisms. Some results are taken from Zhang et al. (2025) and Zadouri et al. (2025). For attention mechanism details, refer to Appendix C.

Method	# Parameters	KV Cache	Loading Per Token Per Device (1 GPU)	Loading Per Token Per Device (2 GPUs)	Loading Per Token Per Device (4 GPUs)	Loading Per Token Per Device (8 GPUs)
MHA (Vaswani et al., 2017)	$4dhd_h$	$2hd_h$	$128d_h$	$64d_h$	$32d_h$	$16d_h$
MQA (Shazeer, 2019)	$2dd_h\,(h+1)$	$2d_h$	$2d_h$	$2d_h$	$2d_h$	$2d_h$
GQA (Ainslie et al., 2023)	$2dd_h\,(h+g)$	$2gd_h$	$16d_h$	$8d_h$	$4d_h$	$2d_h$
MLA (DeepSeek et al., 2024c)	$d'_c\,(d+hd_h+hd_h^R)+dd_h^R$ $+d_c\,(d+2hd_h)+dhd_h$	$d_c+d_h^R$	$4.5d_h$	$4.5d_h$	$4.5d_h$	$4.5d_h$
MFA (Hu et al., 2024)	$d'_c\,(d+h\cdot 2d_h)$ $+2d\cdot 2d_h+dh\cdot 2d_h$	$4d_h$	$4d_h$	$4d_h$	$4d_h$	$4d_h$
TPA (Zhang et al., 2025)	$d(\beta_q+2\beta_{kv})\,(h+d_h)+dhd_h$	$2\beta_{kv}\,(h+d_h)$	$6d_h$	$5d_h$	$4.5d_h$	$4.25d_h$
GLA-2 (Zadouri et al., 2025)	$d'_c\,(d+hd_h+hd_h^R)+dd_h^R$ $+d_c\,(d+hd_h)+dhd_h$	$d_c+d_h^R$	$4.5d_h$	$2.5d_h$	$2.5d_h$	$2.5d_h$
GTA (Zadouri et al., 2025)	$dhd_h+dgd_h+dd_h^R+dhd_h$	$gd_h+d_h^R$	$8.5d_h$	$4.5d_h$	$2.5d_h$	$1.5d_h$
MLRA-2	$d'_c\,(d+hd_h+hd_h^R)+dd_h^R$ $+d_c\,(d+hd_h)+dhd_h$	$d_c+d_h^R$	$4.5d_h$	$2.5d_h$	$1.5d_h$	$1.5d_h$
MLRA-4	$d'_c\,(d+hd_h+hd_h^R)+dd_h^R$ $+d_c\,(d+2hd_h)+dhd_h$	$d_c+d_h^R$	$4.5d_h$	$2.5d_h$	$1.5d_h$	$1.5d_h$

Adopting the variance-calibration strategy from LongCat et al. (2025), we apply analogous rescaling to our MLRA variants. For MLRA-2 and MLRA-4, we rescale the query and KV latent states to ensure that the variance of NoPE queries and keys aligns with that of the partial RoPE components across all branches.

$$\boldsymbol{C}^{\mathrm{Q}} \leftarrow \sqrt{\frac{d}{d'_c}}\,\boldsymbol{C}^{\mathrm{Q}}, \qquad \boldsymbol{C}^{\mathrm{KV}} \leftarrow \sqrt{\frac{4d}{d_c}}\,\boldsymbol{C}^{\mathrm{KV}}. \tag{9}$$

Since summing the attention outputs from multiple branches alters the variance, we apply a rescaling factor to the attention outputs of MLRA-2 and MLRA-4 as follows:

$$\text{(MLRA-2)} \quad \mathbf{O}_{:,i,:} \leftarrow \frac{1}{\sqrt{2}}\,\mathbf{O}_{:,i,:}, \qquad \text{(MLRA-4)} \quad \mathbf{O}_{:,i,:} \leftarrow \frac{1}{2}\,\mathbf{O}_{:,i,:}. \tag{10}$$

Remark 1. *Although these weight matrices are typically initialized with zero mean and a common variance, these conditions are not guaranteed during training. Consequently, Assumption 1 may not strictly hold in practice. However, the effectiveness of this scaling is best assessed through ablation studies, the results of which are detailed in Section 4.2.2.*

3.4 Analysis

KV Cache. We evaluate the per-device KV cache loading under various TP configurations using Qwen3-32B (Yang et al., 2025a) and Kimi-K2 (Team et al., 2025) as base architectures. Qwen3-32B utilizes GQA with 64 query heads and 8 KV heads ($d_h = 128$), setting $g = 8$ KV heads. Kimi-K2 adopts MLA with 64 heads and a partial RoPE dimension ($d_h^R = 64$). For TPA, we maintain the original configuration with $\beta_{kv} = 2$. Table 1 summarizes the per-device KV cache loading under TP as the number of devices increases. To support TP, the official MLA decoding implementation, FlashMLA (Jiashi Li, 2025), distributes up-projection matrices across devices by head. However, this approach leads to redundant KV cache loading; as a result, the per-device loading remains constant at $4.5d_h$ regardless of the TP degree. TPA constructs its h key-value heads as linear combinations of β_{kv} shared heads. It supports TP only for the combination coefficients, while the shared heads must be redundantly loaded by each device. Consequently, the per-device KV cache loading is $4d_h + \frac{2d_h}{\varphi}$, where φ denotes the number of TP devices. GLA-2 partially addresses this by partitioning the latent head into two smaller latent heads, reducing the per-device loading to $2.5d_h$ under 2-way TP. Notably, for MLA with TP > 1 and GLA-2 with TP > 2, the KV cache loading becomes invariant to the number of devices due to sharding constraints, causing the per-device loading to plateau at $4.5d_h$ and $2.5d_h$, respectively. While GQA and GTA require 8-way TP to reduce the per-device loading to $2d_h$ and $1.5d_h$, MLRA achieves $1.5d_h$ with only 4-way TP.

Attention Decoding Arithmetic Intensity. Arithmetic intensity (AI) (Williams et al., 2009), defined as the ratio of floating-point operations to memory access (FLOPs/byte), serves as a critical metric for identifying whether a workload is memory-bound or compute-bound (Zadouri et al., 2025). Given that the context length n is the dominant factor in long-context decoding, we evaluate the arithmetic intensity (AI) of various attention mechanisms, with the results summarized in

Table 2: Comparison of attention decoding arithmetic intensity among attention mechanisms.

Method	MHA	MQA	GQA	MLA	MFA	TPA	GLA-2	GTA	MLRA-2	MLRA-4
Arithmetic Intensity	$\frac{4nhd_h}{4nhd_h}$ ≈ 1	$\frac{4nhd_h}{4nd_h}$ $\approx h$	$\frac{4nhd_h}{4ngd_h}$ $\approx \frac{h}{g}$	$\frac{4nhd_c+2nhd_h^R}{2n\left(d_c+d_h^R\right)}$ $\approx 2h$	$\frac{4nh\cdot 2d_h}{4n\cdot 2d_h}$ $\approx h$	$\frac{4nh\beta_{kv}d_h+4nhd_h}{4n\beta_{kv}(h+d_h)}$ $\approx \frac{(1+\beta_{kv})hd_h}{\beta_{kv}(h+d_h)}$	$\frac{2nh\frac{d_c}{2}+nhd_h^R}{2n\left(\frac{d_c}{2}+d_h^R\right)}$ $\approx h$	$\frac{4nhd_h}{2n\left(gd_h+d_h^R\right)}$ $\approx \frac{2h}{g}$	$\frac{2nh\frac{d_c}{4}+nhd_h^R}{2n\left(\frac{d_c}{4}+d_h^R\right)}$ $\approx h$	$\frac{4nh\frac{d_c}{4}+2nhd_h^R}{2n\left(\frac{d_c}{4}+d_h^R\right)}$ $\approx 2h$

Table 2. MLRA-2 and MLRA-4 achieve AI values of h and $2h$, respectively, maintaining the high arithmetic intensity characteristic of MLA and GLA-2. By significantly increasing the compute-to-memory ratio, MLRA shifts the decoding process away from the HBM bandwidth ceiling toward a compute-limited regime.

4 Experiments

4.1 Experimental Setup

Model Configuration. We adopt the Llama-3 (Llama et al., 2024) architecture (Appendix D) and compare MLRA against the following attention mechanism baselines: MHA (Vaswani et al., 2017), MQA (Shazeer, 2019), GQA (Ainslie et al., 2023), MLA (DeepSeek et al., 2024c), MFA (Hu et al., 2024), TPA (Zhang et al., 2025), GLA-2 (Zadouri et al., 2025), GLA-4, and GTA (Zadouri et al., 2025). GLA-4 compresses the KV cache into four latent heads. We initialize the MHA baseline with the Llama3.2-3B (Llama et al., 2024) configuration and use it as our parameter-count reference. Following Zadouri et al. (2025), for each other attention variant, we adjust the Feed-Forward Network (FFN) intermediate dimension to match the total number of parameters of this MHA baseline. Full details of the architectural hyperparameters are provided in Appendix F.1. All models are implemented on top of the `nanoGPT` (Karpathy, 2022) codebase.

Pretraining Configuration. We pretrain all models at the 2.9B-parameter scale on FineWeb-Edu-100B (Penedo et al., 2024). Each model is pretrained from scratch on 98.3B tokens, with an additional 0.1B tokens for validation. We use the GPT-2 tokenizer with a vocabulary size of 50,304 and follow the standard GPT-3 pretraining setup. We use AdamW (Loshchilov & Hutter, 2019) as the optimizer with $(\beta_1, \beta_2) = (0.9, 0.95), \epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at 1.0. The learning rate is linearly warmed up for the first 2,000 steps, then annealed with cosine decay (Loshchilov & Hutter, 2017) to 10% of the peak. Peak learning rate is 1.6×10^{-4}. We train with a context length of 2,048 tokens and a global batch size of 480 sequences (983,040 tokens per step, $\approx$1.0M) for 100,000 steps. All models are pretrained on 8 NVIDIA H100 80GB GPUs.

Evaluation Benchmark. In addition to evaluating the perplexity from the FineWeb-Edu validation dataset, we evaluate our models on six additional datasets: Wikipedia, C4 (Raffel et al., 2020), the Pile (Gao et al., 2020), RefinedWeb (Penedo et al., 2023), Cosmopedia (Ben Allal et al., 2024), and FineWeb (Penedo et al., 2024) using 0.1B tokens per dataset. We evaluate zero-shot performance on common-sense reasoning benchmarks, including ARC-Easy (ARC-E) (Yadav et al., 2019), ARC-Challenge (ARC-C), OpenBookQA (Mihaylov et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), and PIQA (Bisk et al., 2020), using the `lm-evaluation-harness` (Gao et al., 2024) package. We report normalized accuracy for ARC-E/C, OpenBookQA, HellaSwag, and PIQA, with standard accuracy for all other tasks.

4.2 Preliminary Ablation Results

4.2.1 Initialization

We follow the GPT (Radford et al., 2018) initialization method, where all model weights are initialized using an $\mathcal{N}(0, \sigma = 0.02)$ distribution. However, TPA employs zero initialization for the output projection parameters of the attention and FFN modules, which is an approach also explored in muP (Yang et al., 2021) and LoRA (Hu et al., 2022). To evaluate these different approaches, we conduct an ablation study comparing zero initialization against $\mathcal{N}(0, \sigma = 0.02)$ for the output projection parameters across all models. It is important to note that for MLA, GLA-2, and GLA-4, we apply scaling as discussed in Section 3.3. As illustrated in Figure 1 and Table 38, the results for

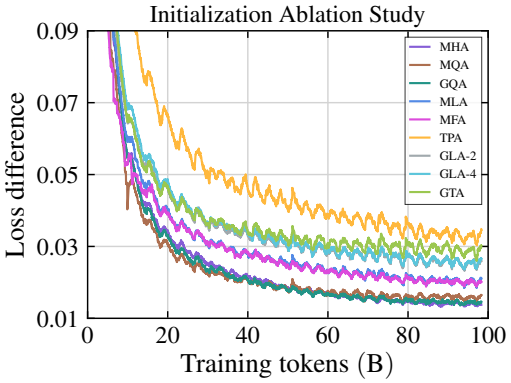

Figure 1: Loss difference between $\mathcal{N}(0, \sigma = 0.02)$ and zero initialization, calculated by subtracting the loss of the latter from the former.

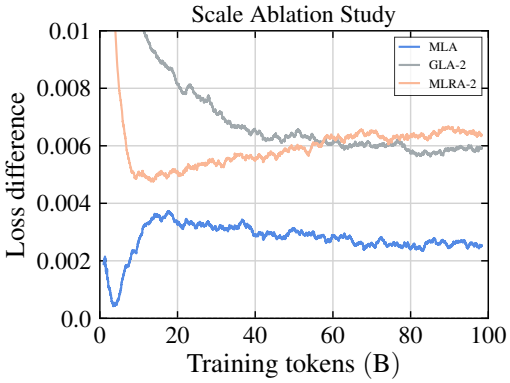

Figure 2: Loss difference between models without and with scaling, calculated by subtracting the loss of the latter from the former.

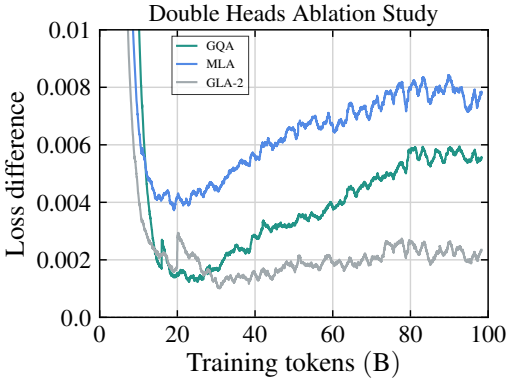

Figure 3: Loss difference between models with and without double heads, calculated by subtracting the loss of the latter from the former.

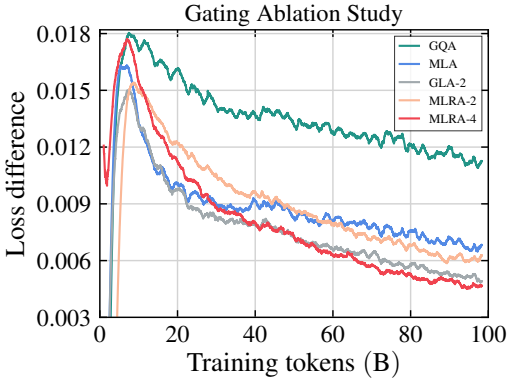

Figure 4: Loss difference between models without and with gating, calculated by subtracting the loss of the latter from the former.

loss and perplexity demonstrate that zero initialization outperforms the $\mathcal{N}(0, \sigma = 0.02)$ distribution. Unless otherwise specified, all models in the following experiments utilize this zero initialization.

4.2.2 Scaling

We evaluate the effectiveness of the scaling on MLA, GLA-2, and MLRA-2. As illustrated in Figure 2, all three models exhibit improved convergence when scaling is applied. As shown in Table 39, all three models achieve lower average perplexity after scaling. Notably, MLA and GLA-2 show more substantial improvements, while MLRA-2 yields a marginal gain. Unless otherwise specified, MLA, GLA, and MLRA in the following experiments utilize this scaling.

4.2.3 Double Heads

While MLRA-2 and MLRA-4 do not increase the number of query heads, their multi-branch design increases the number of attention heads involved in computation. Consequently, we double the number of attention heads for GQA, MLA, and GLA-2 while keeping the KV-cache size fixed to evaluate whether this increase contributes to performance gains. To maintain a constant parameter budget during this adjustment, we reduce the FFN intermediate sizes; the corresponding architectural hyperparameters are detailed in Appendix F.4. As illustrated by the loss curves in Figure 3 and the results in Table 40, doubling the number of attention heads leads to higher loss and fails to decrease perplexity across all three models. These findings suggest that doubling the number of attention heads does not yield any measurable performance improvement. Unless otherwise specified, GQA, MLA, and GLA use the default head count (no head doubling).

Table 3: Validation perplexity (lower is better) across seven datasets: Wikipedia, C4, Pile, RefinedWeb, Cosmopedia, FineWeb, and FineWeb-Edu. The best results are indicated in **bold**, while the second best are underlined.

Method	Wikipedia	C4	Pile	RefinedWeb	Cosmopedia	FineWeb	FineWeb-Edu	Avg
MHA	14.624	16.575	**12.929**	18.698	9.102	15.656	9.434	13.860
MQA	15.134	16.837	14.008	19.202	9.484	15.942	9.533	14.306
GQA	15.057	16.628	13.758	18.885	9.504	15.713	9.427	14.139
MLA	14.567	16.345	12.965	18.523	8.966	15.440	9.284	13.727
MFA	15.693	16.738	13.903	19.125	9.423	15.815	9.506	14.315
TPA	14.789	16.622	13.333	18.971	9.130	15.717	9.333	13.985
GLA-2	14.605	16.323	13.225	18.509	9.118	15.424	9.249	13.779
GLA-4	14.547	16.436	13.229	18.578	9.076	15.535	9.307	13.815
GTA	14.733	16.599	13.402	18.924	9.129	15.672	9.346	13.972
MLRA-2	14.615	16.342	13.236	18.602	9.153	15.439	9.242	13.804
MLRA-4	**14.407**	**16.286**	13.124	**18.398**	**8.937**	**15.361**	**9.193**	**13.672**

Table 4: Downstream evaluation on seven common-sense reasoning benchmarks: ARC-E, ARC-C, OpenBookQA, BoolQ, HellaSwag, Winogrande, and PIQA. ARC-E/C, OpenBookQA, HellaSwag, and PIQA use normalized accuracy (%); others use standard accuracy (%). Best is **bold**; second best is underlined.

Method	ARC-E	ARC-C	OpenBookQA	BoolQ	HellaSwag	Winogrande	PIQA	Avg
MHA	69.11	39.16	40.80	62.26	60.82	57.62	74.86	57.81
MQA	66.16	38.31	41.80	62.05	60.24	59.83	74.48	57.55
GQA	67.13	39.42	42.00	63.39	61.29	56.91	75.08	57.89
MLA	68.22	39.16	42.60	**64.10**	61.39	60.06	**75.68**	58.75
MFA	69.02	39.93	42.40	63.49	60.72	58.96	75.19	58.53
TPA	**69.44**	40.61	41.60	60.03	61.02	57.85	74.54	57.87
GLA-2	68.01	40.19	40.60	63.94	61.54	58.41	75.41	58.30
GLA-4	68.77	41.04	41.20	61.96	61.61	58.09	74.65	58.19
GTA	67.97	39.68	42.60	59.72	61.03	58.48	75.14	57.80
MLRA-2	67.89	**42.24**	42.00	61.65	61.49	59.98	75.52	58.68
MLRA-4	67.63	41.38	**43.00**	61.74	**62.16**	**61.48**	74.48	**58.84**

4.3 Main Results

As shown in Table 3, MLRA-4 achieves the best average perplexity (13.672), outperforming all other models, including MLA (13.727). Notably, MLRA-4 also delivers the lowest perplexity on FineWeb-Edu (9.193). Furthermore, Table 4 demonstrates that MLRA-4 attains the highest average zero-shot accuracy across all common-sense reasoning tasks. This consistent superiority of MLRA-4 over MLRA-2 across both evaluations highlights the benefits of increasing the number of branches.

4.4 Gated Attention

Following Qiu et al. (2025), we introduce a gating mechanism prior to the attention output projection (Appendix E). To maintain a constant parameter budget, we reduce the FFN intermediate size accordingly; detailed architectural hyperparameters are provided in Appendix F.5. As illustrated in Figure 4, all five models exhibit improved convergence with gating applied. As shown in Table 5, gating consistently improves perplexity across all models, with MLRA-4 achieving the best overall average perplexity and MLRA-2 attaining performance comparable to MLA (13.651 vs. 13.642).

4.5 Decoding Efficiency

Decoding Speed. We benchmark long-context attention decoding speed for GQA, MLA, GLA-2, and MLRA-4 on an NVIDIA H100 80GB GPU. For MLA, GLA-2, and MLRA-4, we follow the attention decoding formulation in Eq. (1). All models use 64 heads with a head dimension of 128; for MLA, GLA-2, and MLRA-4, the partial RoPE dimension is 64. MLA is evaluated using DeepSeek's official implementation FlashMLA (Jiashi Li, 2025). GQA and GLA-2 use FlashAttention-3 kernels (Dao et al., 2022; Dao, 2024; Shah et al., 2024). We implement our MLRA-4 kernel based on FlashAttention-3. We evaluate decoding speed across sequence lengths from 131,072 to 2,097,152

Table 5: Validation perplexity (lower is better) w/ gating across seven datasets. The best results are indicated in **bold**, while the second best are underlined.

Method	Wikipedia	C4	Pile	RefinedWeb	Cosmopedia	FineWeb	FineWeb-Edu	Avg
GQA w/ gating	14.362	16.484	13.113	18.696	9.098	15.581	9.311	13.806
MLA w/ gating	**14.346**	16.297	**12.866**	18.456	8.936	15.383	9.212	13.642
GLA-2 w/ gating	14.597	16.286	12.997	18.473	8.986	15.369	9.198	13.701
MLRA-2 w/ gating	14.424	16.252	13.017	18.407	8.924	15.351	9.180	13.651
MLRA-4 w/ gating	14.431	**16.170**	13.073	**18.386**	**8.874**	**15.266**	**9.148**	**13.621**

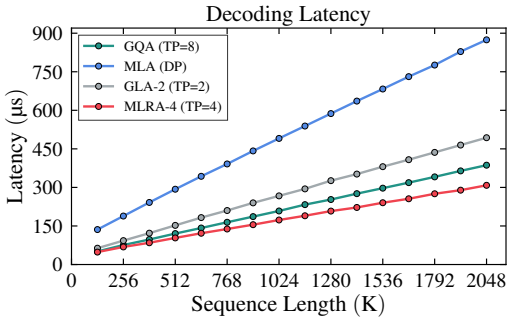

Figure 5: Decoding latency (lower is better) versus sequence length (batch=1) for GQA, MLA, GLA-2, and MLRA-4.

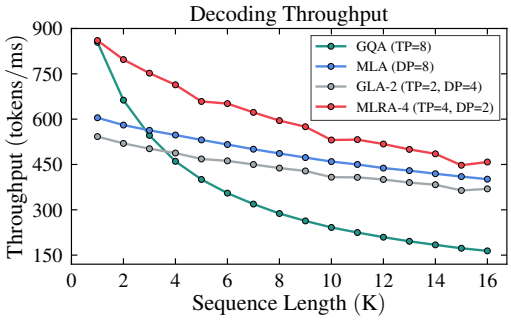

Figure 6: Decoding throughput versus sequence length (batch=128) for GQA, MLA, GLA-2, and MLRA-4.

tokens (128K–2M). As shown in Figure 5, MLRA-4 consistently outperforms all baselines at every length, yielding 1.05×–1.26× speedups over GQA. The gap grows with context length against GQA and GLA-2, while the speedup over MLA remains steady at about 2.8×, indicating that MLRA-4 with TP=4 substantially reduces long-context decoding latency.

Decoding Throughput. We evaluate decoding throughput for GQA, MLA, GLA-2, and MLRA-4 on eight NVIDIA H100 80GB GPUs, fixing the number of attention heads to 128 and the hidden size to 7168, following DeepSeekV3 (DeepSeek et al., 2024a). We set $g = 16$ for GQA. For MLA decoding deployment, there is a trade-off between data parallelism (DP) and tensor parallelism. With DP, we assign different requests to different devices, so attention parameters are replicated across devices and the load can become imbalanced due to varying sequence lengths. With TP, the up-projection parameters are sharded by head, but the KV cache loading is repeated across devices. Following SGLang (Zheng et al., 2024), we aim to eliminate redundant KV cache loading. Therefore, we use DP=8 for MLA, TP=2/DP=4 for GLA-2, TP=4/DP=2 for MLRA-4, and TP=8 for GQA. Throughput is reported for sequence lengths ranging from 1,024 to 16,384 tokens, and our end-to-end measurements include both the pre-attention stage that prepares inputs for the attention kernel and the attention computation itself. We accelerate pre-attention computation with torch.compile (Paszke et al., 2019) for MLA, GLA-2, and MLRA-4, and with custom Triton kernels for GQA. As shown in Figure 6, MLRA-4 achieves the highest decoding throughput across both short and long sequence lengths. This suggests that MLRA-4 with TP=4/DP=2 reduces parameter redundancy relative to MLA's DP=8, while introducing only modest partial RoPE duplication, thereby yielding higher throughput than MLA. For short sequences, GQA outperforms MLA and GLA-2 because pre-attention dominates latency. However, MLRA-4 remains competitive with GQA due to having even fewer query, key, and value parameters, as shown in Appendix F.1.

5 Conclusion

We propose Multi-Head Low-Rank Attention (MLRA), a novel attention mechanism with native 4-way tensor parallelism support. At the 2.9B scale, MLRA-4 achieves state-of-the-art performance on perplexity and zero-shot common-sense reasoning benchmarks. Furthermore, MLRA achieves the lowest decoding latency for long-context sequences (up to 2M tokens) and the highest throughput across sequence lengths from 1K to 16K tokens with 4-way tensor parallelism.

ACKNOWLEDGEMENT

We thank Songlin Yang for helpful discussion. We thank all the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Empirical Methods in Natural Language Processing*, 2023.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. In *Advances in Neural Information Processing Systems*, 2023.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. In *Advances in Neural Information Processing Systems*, 2024.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Cosmopedia. https://huggingface.co/datasets/HuggingFaceTB/cosmopedia, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. Lococo: Dropping in convolutions for long context compression. In *International Conference on Machine Learning*, 2024.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2025.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. Arkvale: Efficient generative LLM inference with recallable key-value eviction. In *Advances in Neural Information Processing Systems*, 2024a.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongloRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, 2024b.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *North American Association for Computational Linguistics*, 2019.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

DeepSeek et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

DeepSeek et al. Deepseek-v3 technical report. https://github.com/deepseek-ai/DeepSeek-V3, 2024b.

DeepSeek et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024c.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *International Midwest Symposium on Circuits and Systems*, 2017.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. In *International Conference on Machine Learning*, 2024.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, 2020.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2024.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Jiaao He and Jidong Zhai. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines. *arXiv preprint arXiv:2403.11421*, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, 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*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jingcheng Hu, Houyi Li, Yinmin Zhang, Zili Wang, Shuigeng Zhou, Xiangyu Zhang, Heung-Yeung Shum, and Daxin Jiang. Multi-matrix factorization attention. *arXiv preprint arXiv:2412.19255*, 2024.

Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. In *Machine Learning and Systems*, 2021.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, 2024.

Shengyu Liu Jiashi Li. Flashmla: Efficient mla decoding kernels. https://github.com/deepseek-ai/FlashMLA, 2025.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, 2020.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *International Conference on Learning Representations*, 2024.

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

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReloRA: High-rank training through low-rank updates. In *International Conference on Learning Representations*, 2024.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. ModeGPT: Modular decomposition for large language model compression. In *International Conference on Learning Representations*, 2025.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: Kv cache compression and streaming for fast large language model serving. In *ACM Special Interest Group on Data Communication*, 2024a.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, 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*, 2024b.

Llama et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Meituan LongCat et al. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*, 2025.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, 2023.

Fanxu Meng, Pingzhi Tang, Xiaojuan Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. Transmla: Multi-head latent attention is all you need. In *Advances in Neural Information Processing Systems*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Empirical Methods in Natural Language Processing*, 2018.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *International Conference on Machine Learning*, 2024.

OpenAI et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: Outperforming curated corpora with web data only. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, 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. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran GV, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. In *Conference on Language Modeling*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Advances in Neural Information Processing Systems*, 2023.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, 2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *International Conference on Learning Representations*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, 2024.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

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

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *International Conference on Machine Learning*, 2025.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. In *Advances in Neural Information Processing Systems*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2024.

Xiaojuan Tang, Fanxu Meng, Pingzhi Tang, Yuxuan Wang, Di Yin, Xing Sun, and Muhan Zhang. Tpla: Tensor parallel latent attention for efficient disaggregated prefill & decode inference. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2025.

Kimi Team et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

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

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, 2018.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 2009.

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

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Empirical Methods in Natural Language Processing*, 2019.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Bowen Yang, Bharat Venkitesh, Dwaraknath Gnaneshwar, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. Rope to nope and back again: A new hybrid attention strategy. In *Advances in Neural Information Processing Systems*, 2025b.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, 2021.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025c.

Ted Zadouri, Hubert Strauss, and Tri Dao. Hardware-efficient attention for fast decoding. In *Conference on Language Modeling*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Association for Computational Linguistics*, 2019.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *International Conference on Learning Representations*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023a.

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

Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *Advances in Neural Information Processing Systems*, 2024a.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient LLMs inference. In *International Conference on Machine Learning*, 2024b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, 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*, 2023b.

Chuanyang Zheng, Jiankai Sun, Yihang Gao, Yuehao Wang, Peihao Wang, Jing Xiong, Liliang Ren, Hao Cheng, Janardhan Kulkarni, Yelong Shen, Atlas Wang, Mac Schwager, Anderson Schneider, Xiaodong Liu, and Jianfeng Gao. Sas: Simulated attention score. In *Advances in Neural Information Processing Systems*, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez De Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. In *International Conference on Machine Learning*, 2024.

Appendix

A NOTATION

Table 6: Notation used throughout this paper.

Symbol	Shape / Type	Meaning
n	scalar	Sequence length (number of tokens).
d	scalar	Model/hidden dimension.
h	scalar	Number of attention heads.
d_h	scalar	Per-head dimension.
d_h^R	scalar	Partial RoPE dimension.
d_r	scalar	RoPE rotation dimension in Theorem 1 (even); typically $d_r = d_h$ or $d_r = d_h^R$.
d_f	scalar	MLP intermediate (FFN) dimension.
g	scalar	Number of groups (or KV heads in MQA/GQA).
r	scalar	Repeat factor $r = h/g$ when g KV heads are broadcast to h query heads.
$\alpha_q, \alpha_{kv}, \alpha_{attn}$	scalar	Variance-calibrating rescaling factors for query/KV latents and attention outputs.
β_q, β_{kv}	scalar	Number of low-rank components in TPA for queries / keys-values.
d_c	scalar	Latent KV dimension in MLA/GLA.
d'_c	scalar	Latent Q dimension in MLA/GLA/MFA.
s	integer	Translation offset.
t_q, t_k	integer	Query/key token positions.
b	integer	Block index.
i	integer	Head index ($i \in \{0, \ldots, h-1\}$).
j	integer	Group index ($j \in \{0, \ldots, g-1\}$).
$\gamma(i)$	integer	Mapping from head index to group index.
$\bar{i}$	integer	Head index within group in GLA-2, $\bar{i} = i - \gamma(i)h/2$.
τ	scalar	Softmax scaling factor, $\tau = 1/\sqrt{d_h + d_h^R}$.
φ	integer	Number of tensor-parallel devices.
$\mathrm{RoPE}(\cdot)$	function	Rotary Position Embedding applied to vectors (implemented via rotation matrices).
$\mathrm{RMSNorm}(\cdot)$	function	Root-mean-square normalization.
$\mathrm{Reshape}(\cdot)$	operator	Tensor reshaping (no data change).
$\mathrm{RepeatInterleave}(\cdot)$	operator	Replication along the head dimension (e.g., broadcasting g KV heads to h heads).
$\mathrm{Concat}(\cdot)$	operator	Concatenation along the last dimension unless stated otherwise.

B THEOREM

B.1 TRANSLATION EQUIVARIANCE

Equivariance is a fundamental property in geometric systems such as molecules, where vector features such as atomic forces or dipole moments must transform consistently with the coordinate system (Weiler et al., 2018; Fuchs et al., 2020; Satorras et al., 2021). In the context of sequence models, a common transformation is sequence translation. Let $\boldsymbol{X} = \left(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n-1)}\right) \in \mathbb{X}$ be a sequence of tokens, and define a translation operator $T_s : \mathbb{X} \to \mathbb{X}$ that translates the entire sequence by s positions. Let $\phi : \mathbb{X} \to \mathbb{Y}$ be a function that maps a sequence to a matrix of attention scores $\phi(\boldsymbol{X}) \in \mathbb{R}^{n \times n}$, where each element $\phi(\boldsymbol{X})_{t_q, t_k} = A\left(\boldsymbol{x}^{(t_q)}, \boldsymbol{x}^{(t_k)}\right)$ denotes the attention score between tokens $\boldsymbol{x}^{(t_q)}$ and $\boldsymbol{x}^{(t_k)}$. We say that ϕ is *translation equivariant* if there exists a corresponding output-space transformation $S_s : \mathbb{Y} \to \mathbb{Y}$ such that:

$$\phi(T_s(\boldsymbol{X})) = S_s(\phi(\boldsymbol{X})), \quad \forall s. \tag{11}$$

This property ensures that the attention score between two tokens depends only on their relative positions, not their absolute positions. That is crucial for batch inference using left padding, where

sequences of different lengths are offset to align ends. The first non-padding token of a sequence is no longer at position 0, yet attention scores remain equivariant to this translation.

B.2 Rotary Position Embedding

Rotary Position Embedding (RoPE) (Su et al., 2024) is a positional encoding method designed to incorporate relative position information directly into the attention mechanism. We show in this section that RoPE is translation equivariant.

Theorem 1. *Given two tokens with query $\boldsymbol{q}$ and key $\boldsymbol{k}$ at positions t_q and t_k, respectively, let $\mathrm{RoPE}(\boldsymbol{q}, t_q)$ and $\mathrm{RoPE}(\boldsymbol{k}, t_k)$ denote the RoPE-encoded vectors. We show that translating both positions by an offset s leaves the inner product unchanged:*

$$\langle \mathrm{RoPE}(\boldsymbol{q}, t_q + s), \mathrm{RoPE}(\boldsymbol{k}, t_k + s) \rangle = \langle \mathrm{RoPE}(\boldsymbol{q}, t_q), \mathrm{RoPE}(\boldsymbol{k}, t_k) \rangle .$$

Equivalently, for the attention-score matrix $\phi(\boldsymbol{X}) \in \mathbb{R}^{n \times n}$ induced by RoPE-based dot products, ϕ is translation equivariant in the sense of Eq. (11) with S_s being the simultaneous row/column shift operator.

Proof. Let the RoPE dimension be d_r (assumed even). RoPE applies a block-diagonal rotation matrix $\boldsymbol{R}_t \in \mathbb{R}^{d_r \times d_r}$ at position t. Writing RoPE as right-multiplication on row vectors, we compute the inner product under RoPE as:

$$\left(\boldsymbol{q}\boldsymbol{R}_{t_q}\right)\left(\boldsymbol{k}\boldsymbol{R}_{t_k}\right)^\top = \boldsymbol{q}\boldsymbol{R}_{t_q}\boldsymbol{R}_{t_k}^\top\boldsymbol{k}^\top = \mathrm{Re}\left[\sum_{\ell=0}^{d_r/2-1} \boldsymbol{q}_{[2\ell:2\ell+2]}\, \boldsymbol{k}^*_{[2\ell:2\ell+2]}\, e^{\mathrm{i}(t_q - t_k)\theta_\ell}\right],$$

where θ_ℓ is the angular frequency for the ℓ-th 2D block and $(\cdot)^*$ denotes complex conjugation under the standard $\mathbb{R}^2 \simeq \mathbb{C}$ identification.

Now consider translating both tokens by an offset s. The relative displacement is unchanged:

$$(t_q + s) - (t_k + s) = t_q - t_k,$$

so the factor $e^{\mathrm{i}(t_q - t_k)\theta_\ell}$ remains unchanged for every ℓ. Therefore,

$$\langle \mathrm{RoPE}(\boldsymbol{q}, t_q + s), \mathrm{RoPE}(\boldsymbol{k}, t_k + s) \rangle = \langle \mathrm{RoPE}(\boldsymbol{q}, t_q), \mathrm{RoPE}(\boldsymbol{k}, t_k) \rangle .$$

This proves RoPE-induced dot-product scores satisfy translation equivariance in Eq. (11). □

Remark 2. *While RoPE preserves dot-product translation equivariance, applying an* arbitrary *linear map after RoPE generally breaks this property. Specifically, consider:*

$$\left\langle \mathrm{RoPE}(\boldsymbol{q}, t_q)\, \boldsymbol{W}^Q,\ \mathrm{RoPE}(\boldsymbol{k}, t_k)\, \boldsymbol{W}^K \right\rangle = \boldsymbol{q}\boldsymbol{R}_{t_q}\boldsymbol{W}^Q\left(\boldsymbol{W}^K\right)^\top \boldsymbol{R}_{t_k}^\top\boldsymbol{k}^\top.$$

The term $\boldsymbol{W}^Q\left(\boldsymbol{W}^K\right)^\top$ breaks translation equivariance by disrupting the expression's dependence on the relative position $t_q - t_k$. The property would only be preserved in the specific case where $\boldsymbol{W}^Q\left(\boldsymbol{W}^K\right)^\top = \boldsymbol{I}$, which would reduce the expression to its original form. However, since this constraint is difficult to enforce during training, translation equivariance is generally lost when applying a linear projection after RoPE.

C Attention Mechanism

C.1 Multi-Head Attention (MHA)

Consider a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n \times d}$. We first project these hidden states into queries, keys, and values using projection matrices $\boldsymbol{W}^{\mathrm{Q}}, \boldsymbol{W}^{\mathrm{K}}, \boldsymbol{W}^{\mathrm{V}} \in \mathbb{R}^{d \times (hd_h)}$:

$$\overline{\boldsymbol{Q}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{Q}}\right), \quad \overline{\boldsymbol{K}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{K}}\right), \quad \overline{\boldsymbol{V}} = \boldsymbol{H}\boldsymbol{W}^{\mathrm{V}},$$

where $\overline{\boldsymbol{Q}}, \overline{\boldsymbol{K}}, \overline{\boldsymbol{V}} \in \mathbb{R}^{n \times (hd_h)}$, h is the number of attention heads, and d_h is the dimensionality of each head. Next, we reshape these matrices to separate the head dimension:

$$\mathbf{Q} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}, [n, h, d_h]\right), \quad \mathbf{K}^{\mathrm{C}} = \mathrm{Reshape}\left(\overline{\boldsymbol{K}}, [n, h, d_h]\right), \quad \mathbf{V}^{\mathrm{C}} = \mathrm{Reshape}\left(\overline{\boldsymbol{V}}, [n, h, d_h]\right),$$

such that $\mathbf{Q}, \mathbf{K}^{\mathrm{C}}, \mathbf{V}^{\mathrm{C}} \in \mathbb{R}^{n \times h \times d_h}$. We cache $\mathbf{K}^{\mathrm{C}}$ and $\mathbf{V}^{\mathrm{C}}$ to accelerate inference.

Remark 3. *Let $\mathbf{Q}, \mathbf{K}^C \in \mathbb{R}^{n\times h\times d_h}$ denote the RoPE-encoded queries and keys after projection and reshaping. For head i, define:*

$$\mathbf{Q}_{t_q,i,:} := \mathbf{Q}\left[t_q,\, i,\, :\right], \quad \mathbf{K}_{t_k,i,:} := \mathbf{K}^C\left[t_k,\, i,\, :\right].$$

Then, for any translation offset s, it follows from Theorem 1 that:

$$\left\langle \mathbf{Q}_{t_q+s,i,:},\, \mathbf{K}_{t_k+s,i,:} \right\rangle = \left\langle \mathbf{Q}_{t_q,i,:},\, \mathbf{K}_{t_k,i,:} \right\rangle.$$

C.2 Multi-Query Attention (MQA) and Grouped-Query Attention (GQA)

Both MQA and GQA reduce the number of key and value heads compared to MHA, while maintaining the full number of query heads. MQA takes this to the extreme by using a single key-value head for all query heads, whereas GQA partitions the query heads into groups that each share a key-value head. Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, the queries are computed using the same projection as in MHA. To reduce KV cache, we use projection matrices $\boldsymbol{W}^{\mathrm{K}}, \boldsymbol{W}^{\mathrm{V}} \in \mathbb{R}^{d\times d_h g}$, where $g < h$ (e.g., $g = 1$ for MQA), to compute:

$$\overline{\boldsymbol{K}}^{\mathrm{C}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{K}}\right), \quad \overline{\boldsymbol{V}}^{\mathrm{C}} = \boldsymbol{H}\boldsymbol{W}^{\mathrm{V}}.$$

These are then reshaped into per-head form:

$$\mathbf{K}^{\mathrm{C}} = \mathrm{Reshape}\left(\overline{\boldsymbol{K}}^{\mathrm{C}},\, [n,\, g,\, d_h]\right), \quad \mathbf{V}^{\mathrm{C}} = \mathrm{Reshape}\left(\overline{\boldsymbol{V}}^{\mathrm{C}},\, [n,\, g,\, d_h]\right).$$

We cache $\mathbf{K}^{\mathrm{C}}$ and $\mathbf{V}^{\mathrm{C}}$ during inference. We repeat them by a factor of $r = h/g$ along the head axis to match the h query heads:

$$\mathbf{K} = \mathrm{RepeatInterleave}\left(\mathbf{K}^{\mathrm{C}},\, r,\, \mathrm{dim} = 1\right) \in \mathbb{R}^{n\times h\times d_h},$$
$$\mathbf{V} = \mathrm{RepeatInterleave}\left(\mathbf{V}^{\mathrm{C}},\, r,\, \mathrm{dim} = 1\right) \in \mathbb{R}^{n\times h\times d_h}.$$

Remark 4. *Let $\mathbf{Q} \in \mathbb{R}^{n\times h\times d_h}$ and $\mathbf{K}^C \in \mathbb{R}^{n\times g\times d_h}$ denote the RoPE-encoded queries and cached keys, respectively. For head i, define:*

$$\mathbf{Q}_{t_q,i,:} := \mathbf{Q}\left[t_q,\, i,\, :\right], \quad \mathbf{K}_{t_k,i,:} := \mathbf{K}^C\left[t_k,\, \left\lfloor \frac{i}{r} \right\rfloor,\, :\right].$$

Since both vectors are RoPE-encoded, for any offset s (with valid indices),

$$\left\langle \mathbf{Q}_{t_q+s,i,:},\, \mathbf{K}_{t_k+s,i,:} \right\rangle = \left\langle \mathbf{Q}_{t_q,i,:},\, \mathbf{K}_{t_k,i,:} \right\rangle.$$

C.3 Multi-Head Latent Attention (MLA)

Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, MLA first computes queries as:

$$\boldsymbol{C}^{\mathrm{Q}} = \alpha_q\, \mathrm{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{DQ}}\right), \quad \overline{\boldsymbol{Q}}^{\mathrm{NoPE}} = \boldsymbol{C}^{\mathrm{Q}}\boldsymbol{W}^{\mathrm{UQ}}, \quad \overline{\boldsymbol{Q}}^{\mathrm{RoPE}} = \mathrm{RoPE}\left(\boldsymbol{C}^{\mathrm{Q}}\boldsymbol{W}^{\mathrm{QR}}\right),$$
$$\boldsymbol{W}^{\mathrm{DQ}} \in \mathbb{R}^{d\times d'_c}, \quad \boldsymbol{W}^{\mathrm{UQ}} \in \mathbb{R}^{d'_c\times(hd_h)}, \quad \boldsymbol{W}^{\mathrm{QR}} \in \mathbb{R}^{d'_c\times(hd_h^R)}.$$

where $\alpha_q = \sqrt{\frac{d}{d'_c}}$ is the rescaling factor for query states $\boldsymbol{C}^{\mathrm{Q}}$. We then reshape queries to separate heads:

$$\mathbf{Q}^{\mathrm{NoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{NoPE}},\, [n,\, h,\, d_h]\right), \quad \mathbf{Q}^{\mathrm{RoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{RoPE}},\, \left[n,\, h,\, d_h^R\right]\right),$$

where $\mathbf{Q}^{\mathrm{NoPE}} \in \mathbb{R}^{n\times h\times d_h}$ and $\mathbf{Q}^{\mathrm{RoPE}} \in \mathbb{R}^{n\times h\times d_h^R}$. These are concatenated along the last dimension to form the final query:

$$\mathbf{Q} = \mathrm{Concat}\left(\left[\mathbf{Q}^{\mathrm{NoPE}},\, \mathbf{Q}^{\mathrm{RoPE}}\right],\, \mathrm{dim{=}2}\right).$$

To reduce the KV cache, MLA obtains shared compressed KV states via a down-projection:

$$\boldsymbol{C}^{\mathrm{KV}} = \alpha_{kv}\, \mathrm{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{DKV}}\right), \quad \boldsymbol{W}^{\mathrm{DKV}} \in \mathbb{R}^{d\times d_c},$$
$$\boldsymbol{K}^{\mathrm{RoPE}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{KR}}\right), \quad \boldsymbol{W}^{\mathrm{KR}} \in \mathbb{R}^{d\times d_h^R},$$

where $\alpha_{kv} = \sqrt{\frac{d}{d_c}}$. Both $\boldsymbol{C}^{\text{KV}}$ and $\boldsymbol{K}^{\text{RoPE}}$ are cached during inference. MLA computes h keys and values using learnable up-projection matrices:

$$\overline{\boldsymbol{K}}^{\text{NoPE}} = \boldsymbol{C}^{\text{KV}}\boldsymbol{W}^{\text{UK}}, \quad \overline{\boldsymbol{V}} = \boldsymbol{C}^{\text{KV}}\boldsymbol{W}^{\text{UV}}, \quad \boldsymbol{W}^{\text{UK}}, \boldsymbol{W}^{\text{UV}} \in \mathbb{R}^{d_c \times (hd_h)}.$$

These are reshaped into per-head form:

$$\mathbf{K}^{\text{NoPE}} = \text{Reshape}\left(\overline{\boldsymbol{K}}^{\text{NoPE}}, [n, h, d_h]\right), \quad \mathbf{K}^{\text{RoPE}} = \text{Reshape}\left(\boldsymbol{K}^{\text{RoPE}}, [n, 1, d_h^R]\right),$$
$$\mathbf{V} = \text{Reshape}\left(\overline{\boldsymbol{V}}, [n, h, d_h]\right),$$

where $\mathbf{K}^{\text{NoPE}} \in \mathbb{R}^{n \times h \times d_h}$ and $\mathbf{V} \in \mathbb{R}^{n \times h \times d_h}$.

To obtain per-head position-aware keys, MLA repeats the partial RoPE key across heads:

$$\mathbf{K} = \text{Concat}\left(\left[\mathbf{K}^{\text{NoPE}}, \text{RepeatInterleave}\left(\mathbf{K}^{\text{RoPE}}, h, \text{dim=1}\right)\right], \text{dim=2}\right).$$

Remark 5. *We analyze the translation equivariance property of MLA. Let* $\mathbf{Q}_{t_q,i,:} = \text{Concat}\left(\mathbf{Q}^{NoPE}[t_q, i, :], \mathbf{Q}^{RoPE}[t_q, i, :]\right)$ *and* $\mathbf{K}_{t_k,i,:} = \text{Concat}\left(\mathbf{K}^{NoPE}[t_k, i, :], \boldsymbol{K}^{RoPE}[t_k, :]\right)$ *denote the query and key vectors for head i at positions t_q and t_k, respectively. The attention score for this head is given by the inner product:*

$$\left\langle \mathbf{Q}_{t_q,i,:}, \mathbf{K}_{t_k,i,:} \right\rangle = \left\langle \mathbf{Q}^{NoPE}[t_q, i, :], \mathbf{K}^{NoPE}[t_k, i, :] \right\rangle + \left\langle \mathbf{Q}^{RoPE}[t_q, i, :], \mathbf{K}^{RoPE}[t_k, :] \right\rangle.$$

Between the two terms, the second term with RoPE is position-dependent yet translation equivariant, due to Theorem 1. The first term is position-independent and thus unchanged under joint translations of t_q and t_k. Therefore, the attention score $\left\langle \mathbf{Q}_{t_q,i,:}, \mathbf{K}_{t_k,i,:} \right\rangle$ is equivariant to translation in input positions. Although MLA introduces positional inductive bias via partial RoPE and is translation equivariant, we refer to this property as semi-translation equivariance to distinguish it from full RoPE translation equivariance.

C.4 Multi-matrix Factorization Attention (MFA)

Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n \times d}$, MFA uses h query heads but only a single shared key-value head (i.e., $g = 1$). MFA first projects $\boldsymbol{H}$ to a low-rank space and applies RMSNorm:

$$\boldsymbol{C}^{\text{Q}} = \text{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{\text{CQ}}\right), \quad \boldsymbol{W}^{\text{CQ}} \in \mathbb{R}^{d \times d_c'}.$$

It then up-projects to all query heads and applies RoPE:

$$\overline{\boldsymbol{Q}} = \text{RoPE}\left(\boldsymbol{C}^{\text{Q}}\boldsymbol{W}^{\text{UQ}}\right), \quad \boldsymbol{W}^{\text{UQ}} \in \mathbb{R}^{d_c' \times (h \cdot 2d_h)}.$$

Finally, we reshape into per-head form:

$$\mathbf{Q} = \text{Reshape}\left(\overline{\boldsymbol{Q}}, [n, h, 2d_h]\right) \in \mathbb{R}^{n \times h \times 2d_h}.$$

To reduce KV cache, MFA computes only one key head and one value head:

$$\overline{\boldsymbol{K}}^{\text{C}} = \text{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\text{K}}\right), \quad \overline{\boldsymbol{V}}^{\text{C}} = \boldsymbol{H}\boldsymbol{W}^{\text{V}}, \quad \boldsymbol{W}^{\text{K}}, \boldsymbol{W}^{\text{V}} \in \mathbb{R}^{d \times 2d_h},$$

where $\overline{\boldsymbol{K}}^{\text{C}}, \overline{\boldsymbol{V}}^{\text{C}} \in \mathbb{R}^{n \times 2d_h}$. We reshape them as

$$\mathbf{K}^{\text{C}} = \text{Reshape}\left(\overline{\boldsymbol{K}}^{\text{C}}, [n, 1, 2d_h]\right), \quad \mathbf{V}^{\text{C}} = \text{Reshape}\left(\overline{\boldsymbol{V}}^{\text{C}}, [n, 1, 2d_h]\right),$$

and cache $\mathbf{K}^{\text{C}}$ and $\mathbf{V}^{\text{C}}$ during inference. We repeat them along the head axis to match the h query heads:

$$\mathbf{K} = \text{RepeatInterleave}\left(\mathbf{K}^{\text{C}}, h, \text{dim} = 1\right) \in \mathbb{R}^{n \times h \times 2d_h},$$
$$\mathbf{V} = \text{RepeatInterleave}\left(\mathbf{V}^{\text{C}}, h, \text{dim} = 1\right) \in \mathbb{R}^{n \times h \times 2d_h}.$$

The analysis of translation equivariance is similar to that of MQA.

C.5 TENSOR PRODUCT ATTENTION (TPA)

TPA achieves KV cache compression through low-rank factorization. It represents each head's key/value at a token as a low-rank mixture of β_{kv} components: component vectors in $\mathbb{R}^{d_h}$ and head-specific scalar coefficients. During inference, TPA caches the component tensors and coefficient tensors, and computes keys/values on the fly via linear combination.

Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, TPA first computes the query/key/value factors:

$$
\begin{aligned}
\overline{\boldsymbol{Q}}^{\mathrm{A}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{AQ}}, \quad \boldsymbol{W}^{\mathrm{AQ}} \in \mathbb{R}^{d\times(\beta_q h)}, \quad \overline{\boldsymbol{Q}}^{\mathrm{A}} \in \mathbb{R}^{n\times(\beta_q h)},\\
\overline{\boldsymbol{Q}}^{\mathrm{C}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{CQ}}, \quad \boldsymbol{W}^{\mathrm{CQ}} \in \mathbb{R}^{d\times(\beta_q d_h)}, \quad \overline{\boldsymbol{Q}}^{\mathrm{C}} \in \mathbb{R}^{n\times(\beta_q d_h)},\\
\overline{\boldsymbol{K}}^{\mathrm{A}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{AK}}, \quad \boldsymbol{W}^{\mathrm{AK}} \in \mathbb{R}^{d\times(\beta_{kv} h)}, \quad \overline{\boldsymbol{K}}^{\mathrm{A}} \in \mathbb{R}^{n\times(\beta_{kv} h)},\\
\overline{\boldsymbol{K}}^{\mathrm{C}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{CK}}, \quad \boldsymbol{W}^{\mathrm{CK}} \in \mathbb{R}^{d\times(\beta_{kv} d_h)}, \quad \overline{\boldsymbol{K}}^{\mathrm{C}} \in \mathbb{R}^{n\times(\beta_{kv} d_h)},\\
\overline{\boldsymbol{V}}^{\mathrm{A}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{AV}}, \quad \boldsymbol{W}^{\mathrm{AV}} \in \mathbb{R}^{d\times(\beta_{kv} h)}, \quad \overline{\boldsymbol{V}}^{\mathrm{A}} \in \mathbb{R}^{n\times(\beta_{kv} h)},\\
\overline{\boldsymbol{V}}^{\mathrm{C}} &= \boldsymbol{H}\boldsymbol{W}^{\mathrm{CV}}, \quad \boldsymbol{W}^{\mathrm{CV}} \in \mathbb{R}^{d\times(\beta_{kv} d_h)}, \quad \overline{\boldsymbol{V}}^{\mathrm{C}} \in \mathbb{R}^{n\times(\beta_{kv} d_h)}.
\end{aligned}
$$

We reshape the projections into 3D tensors:

$$
\begin{aligned}
\mathbf{Q}^{\mathrm{A}} &= \mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{A}}, [n, \beta_q, h]\right),\\
\mathbf{Q}^{\mathrm{C}} &= \mathrm{RoPE}\left(\mathrm{Reshape}\left(\overline{\boldsymbol{Q}}^{\mathrm{C}}, [n, \beta_q, d_h]\right)\right),\\
\mathbf{K}^{\mathrm{A}} &= \mathrm{Reshape}\left(\overline{\boldsymbol{K}}^{\mathrm{A}}, [n, \beta_{kv}, h]\right),\\
\mathbf{K}^{\mathrm{C}} &= \mathrm{RoPE}\left(\mathrm{Reshape}\left(\overline{\boldsymbol{K}}^{\mathrm{C}}, [n, \beta_{kv}, d_h]\right)\right),\\
\mathbf{V}^{\mathrm{A}} &= \mathrm{Reshape}\left(\overline{\boldsymbol{V}}^{\mathrm{A}}, [n, \beta_{kv}, h]\right),\\
\mathbf{V}^{\mathrm{C}} &= \mathrm{Reshape}\left(\overline{\boldsymbol{V}}^{\mathrm{C}}, [n, \beta_{kv}, d_h]\right),
\end{aligned}
$$

so that $\mathbf{Q}^{\mathrm{A}} \in \mathbb{R}^{n\times\beta_q\times h}$, $\mathbf{Q}^{\mathrm{C}} \in \mathbb{R}^{n\times\beta_q\times d_h}$, $\mathbf{K}^{\mathrm{A}} \in \mathbb{R}^{n\times\beta_{kv}\times h}$, $\mathbf{K}^{\mathrm{C}} \in \mathbb{R}^{n\times\beta_{kv}\times d_h}$, $\mathbf{V}^{\mathrm{A}} \in \mathbb{R}^{n\times\beta_{kv}\times h}$, $\mathbf{V}^{\mathrm{C}} \in \mathbb{R}^{n\times\beta_{kv}\times d_h}$.

For each token position $t \in \{0, \ldots, n-1\}$, the final query, key, and value matrices are computed as:

$$
\begin{aligned}
\mathbf{Q}\,[t, :, :] &= \frac{1}{\beta_q}\left(\mathbf{Q}^{\mathrm{A}}\,[t, :, :]\right)^{\top}\mathbf{Q}^{\mathrm{C}}\,[t, :, :] \in \mathbb{R}^{h\times d_h},\\
\mathbf{K}\,[t, :, :] &= \frac{1}{\beta_{kv}}\left(\mathbf{K}^{\mathrm{A}}\,[t, :, :]\right)^{\top}\mathbf{K}^{\mathrm{C}}\,[t, :, :] \in \mathbb{R}^{h\times d_h},\\
\mathbf{V}\,[t, :, :] &= \frac{1}{\beta_{kv}}\left(\mathbf{V}^{\mathrm{A}}\,[t, :, :]\right)^{\top}\mathbf{V}^{\mathrm{C}}\,[t, :, :] \in \mathbb{R}^{h\times d_h}.
\end{aligned}
$$

During inference, TPA caches $\mathbf{K}^{\mathrm{A}}, \mathbf{K}^{\mathrm{C}}, \mathbf{V}^{\mathrm{A}}, \mathbf{V}^{\mathrm{C}}$.

Remark 6. *Fix a head index i and token positions t_q, t_k. Let $\mathbf{Q}_{t_q,i,:} := \mathbf{Q}\,[t_q, i, :] \in \mathbb{R}^{d_h}$ and $\mathbf{K}_{t_k,i,:} := \mathbf{K}\,[t_k, i, :] \in \mathbb{R}^{d_h}$. From the computation above, we have*

$$
\mathbf{Q}_{t_q,i,:} = \frac{1}{\beta_q}\sum_{b_q=0}^{\beta_q-1}\mathbf{Q}^{A}\,[t_q, b_q, i]\;\mathbf{Q}^{C}\,[t_q, b_q, :], \qquad \mathbf{K}_{t_k,i,:} = \frac{1}{\beta_{kv}}\sum_{b_{kv}=0}^{\beta_{kv}-1}\mathbf{K}^{A}\,[t_k, b_{kv}, i]\;\mathbf{K}^{C}\,[t_k, b_{kv}, :].
$$

Therefore, the inner product expands as

$$
\left\langle \mathbf{Q}_{t_q,i,:}, \mathbf{K}_{t_k,i,:}\right\rangle = \frac{1}{\beta_q\beta_{kv}}\sum_{b_q=0}^{\beta_q-1}\sum_{b_{kv}=0}^{\beta_{kv}-1}\mathbf{Q}^{A}\,[t_q, b_q, i]\,\mathbf{K}^{A}\,[t_k, b_{kv}, i]\left\langle \mathbf{Q}^{C}\,[t_q, b_q, :], \mathbf{K}^{C}\,[t_k, b_{kv}, :]\right\rangle.
$$

Since $\mathbf{Q}^C$ *and* $\mathbf{K}^C$ *are RoPE-encoded, Theorem 1 implies that for any offset* s*,*

$$\left\langle \mathbf{Q}^C\left[t_q+s,\, b_q,\, :\right],\, \mathbf{K}^C\left[t_k+s,\, b_{kv},\, :\right]\right\rangle = \left\langle \mathbf{Q}^C\left[t_q,\, b_q,\, :\right],\, \mathbf{K}^C\left[t_k,\, b_{kv},\, :\right]\right\rangle, \quad \forall\, b_q,\, b_{kv}.$$

Because the scalar coefficients $\mathbf{Q}^A\left[t_q,\, b_q,\, i\right]$ *and* $\mathbf{K}^A\left[t_k,\, b_{kv},\, i\right]$ *shift with the tokens under translation, the full double-sum is equivariant under jointly translating* t_q *and* t_k *by the same offset* s*:*

$$\left\langle \mathbf{Q}_{t_q+s,i,:},\, \mathbf{K}_{t_k+s,i,:}\right\rangle = \left\langle \mathbf{Q}_{t_q,i,:},\, \mathbf{K}_{t_k,i,:}\right\rangle.$$

Thus, TPA preserves translation equivariance of attention scores with RoPE.

C.6 Grouped Latent Attention (GLA)

Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, GLA divides the h attention heads into g groups (e.g., $g=2$), where each group has $r = h/g$ heads. GLA adopts the same query computation mechanism as MLA.

Instead of a single compressed KV state, GLA computes g independent compressed states:

$$\boldsymbol{C}^{j,\mathrm{KV}} = \alpha_{kv}\,\mathrm{RMSNorm}\left(\boldsymbol{H}\boldsymbol{W}^{j,\mathrm{DKV}}\right), \quad \boldsymbol{W}^{j,\mathrm{DKV}} \in \mathbb{R}^{d\times(d_c/g)},$$

where $j \in \{0,\ldots,g-1\}$, d_c is the total latent dimension, each group uses d_c/g dimension, and $\alpha_{kv} = \sqrt{\frac{gd}{d_c}}$.

The RoPE keys remain shared across all groups:

$$\boldsymbol{K}^{\mathrm{RoPE}} = \mathrm{RoPE}\left(\boldsymbol{H}\boldsymbol{W}^{\mathrm{KR}}\right), \quad \boldsymbol{W}^{\mathrm{KR}} \in \mathbb{R}^{d\times d_h^R}.$$

During inference, we cache $\left\{\boldsymbol{C}^{j,\mathrm{KV}},\ldots,\boldsymbol{C}^{g-1,\mathrm{KV}}\right\}$ and $\boldsymbol{K}^{\mathrm{RoPE}}$ with total KV cache size of $d_c + d_h^R$ per token.

Each group independently computes its keys and values:

$$\overline{\boldsymbol{K}}^{j,\mathrm{NoPE}} = \boldsymbol{C}^{j,\mathrm{KV}}\boldsymbol{W}^{j,\mathrm{UK}}, \quad \overline{\boldsymbol{V}}^{j} = \boldsymbol{C}^{j,\mathrm{KV}}\boldsymbol{W}^{j,\mathrm{UV}}, \quad \boldsymbol{W}^{j,\mathrm{UK}}, \boldsymbol{W}^{j,\mathrm{UV}} \in \mathbb{R}^{(d_c/g)\times(rd_h)},$$

where $r = h/g$ is the number of heads per group.

Reshape into per-head form for each group:

$$\mathbf{K}^{j,\mathrm{NoPE}} = \mathrm{Reshape}\left(\overline{\boldsymbol{K}}^{j,\mathrm{NoPE}},\, [n,\, r,\, d_h]\right), \quad \mathbf{V}^{j} = \mathrm{Reshape}\left(\overline{\boldsymbol{V}}^{j},\, [n,\, r,\, d_h]\right),$$
$$\mathbf{K}^{\mathrm{RoPE}} = \mathrm{Reshape}\left(\boldsymbol{K}^{\mathrm{RoPE}},\, \left[n,\, 1,\, d_h^R\right]\right)$$

Construct position-aware keys for each group by repeating the shared RoPE keys:

$$\mathbf{K}^{j} = \mathrm{Concat}\left(\left[\mathbf{K}^{j,\mathrm{NoPE}},\, \mathrm{RepeatInterleave}\left(\mathbf{K}^{\mathrm{RoPE}},\, r,\, \text{dim=1}\right)\right],\, \text{dim=2}\right) \in \mathbb{R}^{n\times r\times(d_h+d_h^R)}.$$

We finally concatenate all $\mathbf{K}^j$, $\mathbf{V}^j$ to obtain:

$$\mathbf{K} = \mathrm{Concat}\left(\left[\mathbf{K}^0,\,\ldots,\,\mathbf{K}^{g-1}\right],\, \text{dim=1}\right), \quad \mathbf{V} = \mathrm{Concat}\left(\left[\mathbf{V}^0,\,\ldots,\,\mathbf{V}^{g-1}\right],\, \text{dim=1}\right),$$

where $\mathbf{K} \in \mathbb{R}^{n\times h\times(d_h+d_h^R)}$ and $\mathbf{V} \in \mathbb{R}^{n\times h\times d_h}$. The analysis of translation equivariance is similar to that of MLA.

C.7 Grouped-Tied Attention (GTA)

Given a sequence of n tokens with hidden states $\boldsymbol{H} \in \mathbb{R}^{n\times d}$, GTA uses h query heads and g value heads, and computes queries as

$$\overline{\boldsymbol{Q}} = \boldsymbol{H}\boldsymbol{W}^{\mathrm{Q}}, \quad \boldsymbol{W}^{\mathrm{Q}} \in \mathbb{R}^{d\times(hd_h)}, \quad \widetilde{\mathbf{Q}} = \mathrm{Reshape}\left(\overline{\boldsymbol{Q}},\, [n,\, h,\, d_h]\right) \in \mathbb{R}^{n\times h\times d_h}.$$

We split $\widetilde{Q}$ into NoPE and RoPE parts and apply RoPE to the latter:

$$\mathbf{Q}^{\text{NoPE}} = \widetilde{\mathbf{Q}}\left[:, :, : d_h - d_h^R\right] \in \mathbb{R}^{n \times h \times \left(d_h - d_h^R\right)},$$
$$\mathbf{Q}^{\text{RoPE}} = \text{RoPE}\left(\widetilde{\mathbf{Q}}\left[:, :, d_h - d_h^R :\right]\right) \in \mathbb{R}^{n \times h \times d_h^R},$$
$$\mathbf{Q} = \text{Concat}\left(\left[\mathbf{Q}^{\text{NoPE}}, \mathbf{Q}^{\text{RoPE}}\right], \text{dim=2}\right) \in \mathbb{R}^{n \times h \times d_h}.$$

GTA computes a single RoPE key shared across all heads:

$$\overline{\boldsymbol{K}}^{\text{RoPE}} = \boldsymbol{H}\boldsymbol{W}^{\text{KR}}, \quad \boldsymbol{W}^{\text{KR}} \in \mathbb{R}^{d \times d_h^R}, \quad \boldsymbol{K}^{\text{RoPE}} = \text{RoPE}\left(\overline{\boldsymbol{K}}^{\text{RoPE}}\right),$$
$$\mathbf{K}^{\text{RoPE}} = \text{Reshape}\left(\boldsymbol{K}^{\text{RoPE}}, \left[n, 1, d_h^R\right]\right)$$

To reduce KV cache, GTA computes grouped value states with only g heads:

$$\overline{\boldsymbol{V}} = \boldsymbol{H}\boldsymbol{W}^{\text{KV}}, \quad \boldsymbol{W}^{\text{KV}} \in \mathbb{R}^{d \times (g d_h)}, \quad \mathbf{V}^{\text{C}} = \text{Reshape}\left(\overline{\boldsymbol{V}}, \left[n, g, d_h\right]\right).$$

We cache $\mathbf{V}^{\text{C}}$ and $\boldsymbol{K}^{\text{RoPE}}$ during inference. We repeat $\mathbf{V}^{\text{C}}$ along the head axis with $r = h/g$ to form the final values:

$$\mathbf{V} = \text{RepeatInterleave}\left(\mathbf{V}^{\text{C}}, r, \text{dim} = 1\right) \in \mathbb{R}^{n \times h \times d_h}.$$

We then form the final keys by tying the NoPE part of keys to the values and concatenating with the shared RoPE key:

$$\mathbf{K} = \text{Concat}\left(\left[\mathbf{V}\left[:, :, : d_h - d_h^R\right], \text{RepeatInterleave}\left(\mathbf{K}^{\text{RoPE}}, h, \text{dim=1}\right)\right], \text{dim=2}\right) \in \mathbb{R}^{n \times h \times d_h}.$$

The analysis of translation equivariance is similar to that of MLA.

D LLAMA-3 ARCHITECTURE

Given hidden states $\boldsymbol{H} \in \mathbb{R}^{n \times d}$ for a sequence of n tokens, we first compute the attention output

$$\boldsymbol{H}' = \text{RMSNorm}\left(\boldsymbol{H}\right)$$
$$\boldsymbol{O}^{\text{attn}} = \text{Attention}\left(\boldsymbol{H}'\right) \in \mathbb{R}^{n \times (h d_h)},$$

then project back to the model dimension and add a residual:

$$\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{O}^{\text{attn}}\boldsymbol{W}^{\text{O,attn}}, \qquad \boldsymbol{W}^{\text{O,attn}} \in \mathbb{R}^{(h d_h) \times d}.$$

Next, an MLP block (gated form) is applied:

$$\boldsymbol{H}' = \text{RMSNorm}\left(\boldsymbol{H}\right)$$
$$\boldsymbol{O}^{\text{mlp}} = \sigma\left(\boldsymbol{H}'\boldsymbol{W}^1\right) \odot \left(\boldsymbol{H}'\boldsymbol{W}^2\right), \qquad \boldsymbol{W}^1, \boldsymbol{W}^2 \in \mathbb{R}^{d \times d_f},$$

followed by the output projection and residual:

$$\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{O}^{\text{mlp}}\boldsymbol{W}^{\text{O,mlp}}, \qquad \boldsymbol{W}^{\text{O,mlp}} \in \mathbb{R}^{d_f \times d},$$

where $\sigma(\cdot)$ is an elementwise nonlinearity function such as SiLU and $\odot$ denotes elementwise multiplication.

E GATED ATTENTION

Given hidden states $\boldsymbol{H} \in \mathbb{R}^{n \times d}$ for a sequence of n tokens, we first compute the attention output with gated score

$$\boldsymbol{G} = \varsigma\left(\boldsymbol{H}\boldsymbol{W}^{\text{G}}\right)$$
$$\boldsymbol{H}' = \text{RMSNorm}\left(\boldsymbol{H}\right)$$
$$\boldsymbol{O}^{\text{attn}} = \text{Attention}\left(\boldsymbol{H}'\right) \odot \boldsymbol{G} \in \mathbb{R}^{n \times (h d_h)},$$

then project back to the model dimension and add a residual:

$$\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{O}^{\text{attn}}\boldsymbol{W}^{\text{O,attn}}, \qquad \boldsymbol{W}^{\text{O,attn}} \in \mathbb{R}^{(hd_h)\times d}.$$

Next, an MLP block (gated form) is applied:

$$\boldsymbol{H}' = \text{RMSNorm}\left(\boldsymbol{H}\right)$$
$$\boldsymbol{O}^{\text{mlp}} = \sigma\left(\boldsymbol{H}'\boldsymbol{W}^1\right) \odot \left(\boldsymbol{H}'\boldsymbol{W}^2\right), \qquad \boldsymbol{W}^1, \boldsymbol{W}^2 \in \mathbb{R}^{d\times d_f},$$

followed by the output projection and residual:

$$\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{O}^{\text{mlp}}\boldsymbol{W}^{\text{O,mlp}}, \qquad \boldsymbol{W}^{\text{O,mlp}} \in \mathbb{R}^{d_f\times d},$$

where $\varsigma(\cdot)$ is an elementwise nonlinearity function such as sigmoid and $\odot$ denotes elementwise multiplication.

F Architectural Hyperparameters

F.1 Architectural Hyperparameters for Main Results

Our model is based on the Llama-3 architecture, adopting a configuration largely consistent with Llama-3.2-3B but modified to 24 layers (down from 28). The architecture utilizes 24 attention heads, a model hidden dimension (d) of 3072, a head dimension (d_h) of 128, and an intermediate Feedforward Network (FFN) dimension (d_f) of 8192. The architectural hyperparameters for our baselines are aligned with their original implementations. Specifically, MLA is configured with latent dimensions $d'_c = 12d_h$, $d_c = 4d_h$, and $d_h^R = 0.5d_h$; GLA, along with our proposed MLRA, adopts $d'_c = 8d_h$, $d_c = 4d_h$, and $d_h^R = 0.5d_h$; and TPA uses ranks $\beta_q = 6$ and $\beta_{kv} = 2$. For GQA and GTA, we set the number of KV heads to $g = h/4$. We report the detailed architectural hyperparameters for our main experiments in Tables 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, and 17.

Table 7: Model configuration of MHA for main results.

Model Size	# Parameters	# Layers	h	d	d_h	d_f
2.9B	2872.59M	24	24	3072	128	8192

Table 8: Model configuration of MQA for main results.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.00M	24	24	1	3072	128	10152

Table 9: Model configuration of GQA for main results.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.59M	24	24	6	3072	128	9728

Table 10: Model configuration of MLA for main results.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.05M	24	24	1536	512	3072	$\sqrt{2}$	$\sqrt{6}$	128	64	9448

Table 11: Model configuration of MFA for main results.

Model Size	# Parameters	# Layers	h	g	d'_c	d	d_h	d_f
2.9B	2873.23M	24	24	1	2048	3072	256	8024

Table 12: Model configuration of TPA for main results.

Model Size	# Parameters	# Layers	h	β_q	β_{kv}	d	d_h	d_f
2.9B	2873.18M	24	24	6	2	3072	128	10760

Table 13: Model configuration of GLA-2 for main results.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	2	1024	512	3072	$\sqrt{3}$	$\sqrt{12}$	128	64	10048

Table 14: Model configuration of GLA-4 for main results.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2873.22M	24	24	4	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	128	64	10136

Table 15: Model configuration of GTA for main results.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_h^R	d_f
2.9B	2872.00M	24	24	6	3072	128	64	9960

Table 16: Model configuration of MLRA-2 for main results.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	α_{attn}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	$\frac{\sqrt{2}}{2}$	128	64	10048

Table 17: Model configuration of MLRA-4 for main results.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	α_{attn}	d_h	d_h^R	d_f
2.9B	2873.22M	24	24	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	$\frac{1}{2}$	128	64	9880

F.2 Architectural Hyperparameters for Initialization Ablation Study

In our initialization ablation study, we focus on the initialization of the attention and FFN output projections parameters ($\boldsymbol{W}^{\text{O, attn}}, \boldsymbol{W}^{\text{O, mlp}}$). We evaluate two distinct initialization strategies: zero initialization versus a Gaussian distribution $\mathcal{N}(0, \sigma = 0.02)$, to identify which yields better performance. To isolate the impact of the initialization strategy, the model architecture and all other hyperparameters are kept identical to those used for our main results. We report the detailed architectural hyperparameters for our initialization ablation in Tables 18, 19, 20, 21, 22, 23, 24, 25, and 26.

Table 18: Model configuration of MHA for initialization ablation.

Model Size	# Parameters	# Layers	h	d	d_h	d_f
2.9B	2872.59M	24	24	3072	128	8192

Table 19: Model configuration of MQA for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.00M	24	24	1	3072	128	10152

Table 20: Model configuration of GQA for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.59M	24	24	6	3072	128	9728

Table 21: Model configuration of MLA for initialization ablation.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.05M	24	24	1536	512	3072	$\sqrt{2}$	$\sqrt{6}$	128	64	9448

Table 22: Model configuration of MFA for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d'_c	d	d_h	d_f
2.9B	2873.23M	24	24	1	2048	3072	256	8024

Table 23: Model configuration of TPA for initialization ablation.

Model Size	# Parameters	# Layers	h	β_q	β_{kv}	d	d_h	d_f
2.9B	2873.18M	24	24	6	2	3072	128	10760

Table 24: Model configuration of GLA-2 for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	2	1024	512	3072	$\sqrt{3}$	$\sqrt{12}$	128	64	10048

Table 25: Model configuration of GLA-4 for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2873.22M	24	24	4	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	128	64	10136

Table 26: Model configuration of GTA for initialization ablation.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_h^R	d_f
2.9B	2872.00M	24	24	6	3072	128	64	9960

F.3 Architectural Hyperparameters for Scaling Ablation Study

In our scaling ablation study, we investigate the impact of the scaling factors α_q, α_{kv}, and α_{attn} applied to the query latent states ($\boldsymbol{C}^{\mathrm{Q}}$), the KV latent states ($\boldsymbol{C}^{\mathrm{KV}}$), and the final attention output ($\mathbf{O}$), respectively. To determine the optimal configuration, we compare the model's performance with and without these scaling factors, where the 'without' setting corresponds to fixing α_q, α_{kv}, and α_{attn} to 1. To isolate the impact of this scaling strategy, the model architecture and all other hyperparameters remain identical to those used in our main results. Detailed architectural specifications for these ablation experiments are provided in Tables 27, 28, and 29.

Table 27: Model configuration of MLA in the absence of scaling factors.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.05M	24	24	1536	512	3072	1	1	128	64	9448

Table 28: Model configuration of GLA-2 in the absence of scaling factors.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	2	1024	512	3072	1	1	128	64	10048

Table 29: Model configuration of MLRA-2 in the absence of scaling factors.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	α_{attn}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	1024	512	3072	1	1	1	128	64	10048

F.4 Architectural Hyperparameters for Double Heads Ablation Study

In our head-count ablation study, we investigate whether doubling the number of attention heads for GQA, MLA, and GLA-2 improves performance. Specifically, we increase the number of heads to 48 while maintaining the original KV cache size. To maintain parameter parity with our main results, we decrease the Feed-Forward Network (FFN) intermediate dimension. By keeping all other hyperparameters identical, we isolate the specific impact of the doubled head count. The detailed architectural specifications for these experiments are provided in Tables 30, 31, and 32.

Table 30: Model configuration of GQA parameterized with $2\times$ attention heads.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.59M	24	48	6	3072	128	7680

Table 31: Model configuration of MLA parameterized with $2\times$ attention heads.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2873.23M	24	48	1536	512	3072	$\sqrt{2}$	$\sqrt{6}$	128	64	7320

Table 32: Model configuration of GLA-2 parameterized with $2\times$ attention heads.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2873.22M	24	48	2	1024	512	3072	$\sqrt{3}$	$\sqrt{12}$	128	64	8344

F.5 Architectural Hyperparameters for Gated Attention Study

In our gated attention study, we investigate whether incorporating a gating mechanism (Hochreiter & Schmidhuber, 1997; Srivastava et al., 2015; Dey & Salem, 2017; Qiu et al., 2025) into GQA, MLA, GLA-2, MLRA-2, and MLRA-4 improves performance. Specifically, we integrate gated attention into these architectures as shown in Appendix E. To maintain parameter parity with our main results, we proportionally decrease the Feed-Forward Network (FFN) intermediate dimension to offset the additional gate parameters. By keeping all other hyperparameters identical, we isolate the specific impact of the gating strategy. The detailed architectural specifications for these experiments are provided in Tables 33, 34, 35, 36, and 37.

Table 33: Model configuration of GQA incorporating gated attention.

Model Size	# Parameters	# Layers	h	g	d	d_h	d_f
2.9B	2872.59M	24	24	6	3072	128	8704

Table 34: Model configuration of MLA incorporating gated attention.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.05M	24	24	1536	512	3072	$\sqrt{2}$	$\sqrt{6}$	128	64	8424

Table 35: Model configuration of GLA-2 incorporating gated attention.

Model Size	# Parameters	# Layers	h	g	d'_c	d_c	d	α_q	α_{kv}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	2	1024	512	3072	$\sqrt{3}$	$\sqrt{12}$	128	64	9024

Table 36: Model configuration of MLRA-2 incorporating gated attention.

Model Size	# Parameters	# Layers	h	d'_c	d_c	d	α_q	α_{kv}	α_{attn}	d_h	d_h^R	d_f
2.9B	2872.63M	24	24	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	$\frac{\sqrt{2}}{2}$	128	64	9024

Table 37: Model configuration of MLRA-4 incorporating gated attention.

Model Size	# Parameters	# Layers	h	d_c'	d_c	d	α_q	α_{kv}	α_{attn}	d_h	d_h^R	d_f
2.9B	2873.22M	24	24	1024	512	3072	$\sqrt{3}$	$\sqrt{24}$	$\frac{1}{2}$	128	64	8856

G Additional Experimental Results

Table 38: Validation perplexity (lower is better) across seven datasets: Wikipedia, C4, Pile, RefinedWeb, Cosmopedia, FineWeb, and FineWeb-Edu. We compare two initialization strategies, zero versus Gaussian ($\mathcal{N}(0, \sigma = 0.02)$), applied to the output projection weights $\boldsymbol{W}^{\text{O, attn}}$ and $\boldsymbol{W}^{\text{O, mlp}}$.

Method	Initialization	Wikipedia	C4	Pile	RefinedWeb	Cosmopedia	FineWeb	FineWeb-Edu	Avg
MHA	$\mathcal{N}(0, \sigma = 0.02)$	14.759	16.800	13.282	18.988	9.356	15.904	9.571	14.094
MHA	zero	14.624	16.575	12.929	18.698	9.102	15.656	9.434	13.860
MQA	$\mathcal{N}(0, \sigma = 0.02)$	14.708	17.075	13.500	19.301	9.510	16.190	9.697	14.283
MQA	zero	15.134	16.837	14.008	19.202	9.484	15.942	9.533	14.306
GQA	$\mathcal{N}(0, \sigma = 0.02)$	14.687	16.882	13.528	19.084	9.422	15.974	9.571	14.164
GQA	zero	15.057	16.628	13.758	18.885	9.504	15.713	9.427	14.139
MLA	$\mathcal{N}(0, \sigma = 0.02)$	14.571	16.624	13.113	18.837	9.110	15.740	9.490	13.927
MLA	zero	14.567	16.345	12.965	18.523	8.966	15.440	9.284	13.727
MFA	$\mathcal{N}(0, \sigma = 0.02)$	15.123	17.032	13.752	19.133	9.550	16.138	9.707	14.374
MFA	zero	15.693	16.738	13.903	19.125	9.423	15.815	9.506	14.315
TPA	$\mathcal{N}(0, \sigma = 0.02)$	15.205	17.128	13.814	19.445	9.844	16.227	9.682	14.478
TPA	zero	14.789	16.622	13.333	18.971	9.130	15.717	9.333	13.985
GLA-2	$\mathcal{N}(0, \sigma = 0.02)$	14.717	16.675	13.216	18.886	9.259	15.799	9.510	14.009
GLA-2	zero	14.605	16.323	13.225	18.509	9.118	15.424	9.249	13.779
GLA-4	$\mathcal{N}(0, \sigma = 0.02)$	14.858	16.791	13.522	18.953	9.374	15.914	9.571	14.140
GLA-4	zero	14.547	16.436	13.229	18.578	9.076	15.535	9.307	13.815
GTA	$\mathcal{N}(0, \sigma = 0.02)$	14.896	16.959	13.621	19.277	9.536	16.061	9.647	14.285
GTA	zero	14.733	16.599	13.402	18.924	9.129	15.672	9.346	13.972

Table 39: Validation perplexity (lower is better) across seven datasets: Wikipedia, C4, Pile, RefinedWeb, Cosmopedia, FineWeb, and FineWeb-Edu. This analysis specifically compares models without and with scaling.

Method	Scaling	Wikipedia	C4	Pile	RefinedWeb	Cosmopedia	FineWeb	FineWeb-Edu	Avg
MLA	w/o	14.461	16.386	13.218	18.636	8.961	15.485	9.307	13.779
MLA	w/	14.567	16.345	12.965	18.523	8.966	15.440	9.284	13.727
GLA-2	w/o	14.518	16.467	13.179	18.612	9.138	15.565	9.305	13.827
GLA-2	w/	14.605	16.323	13.225	18.509	9.118	15.424	9.249	13.779
MLRA-2	w/o	14.326	16.485	13.145	18.657	9.168	15.570	9.304	13.808
MLRA-2	w/	14.615	16.342	13.236	18.602	9.153	15.439	9.242	13.804

Table 40: Validation perplexity (lower is better) across seven datasets: Wikipedia, C4, Pile, RefinedWeb, Cosmopedia, FineWeb, and FineWeb-Edu. This analysis specifically compares models with and without $2\times$ attention heads.

Method	$2\times$ Attention Heads	Wikipedia	C4	Pile	RefinedWeb	Cosmopedia	FineWeb	FineWeb-Edu	Avg
GQA	w/	15.280	16.702	13.789	18.961	9.486	15.785	9.490	14.213
GQA	w/o	15.057	16.628	13.758	18.885	9.504	15.713	9.427	14.139
MLA	w/	14.771	16.432	13.108	18.615	9.029	15.529	9.371	13.836
MLA	w/o	14.567	16.345	12.965	18.523	8.966	15.440	9.284	13.727
GLA-2	w/	14.969	16.313	13.428	18.569	8.991	15.410	9.281	13.851
GLA-2	w/o	14.605	16.323	13.225	18.509	9.118	15.424	9.249	13.779

H ILLUSTRATION

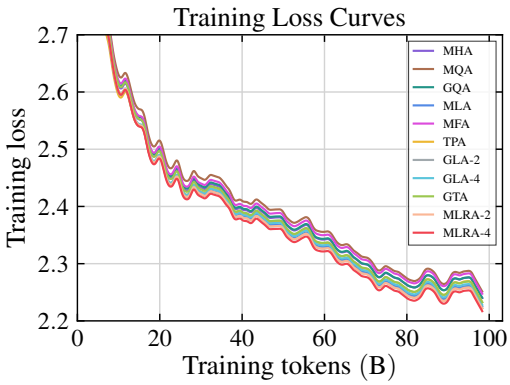

Figure 7: Training loss curves for all models.

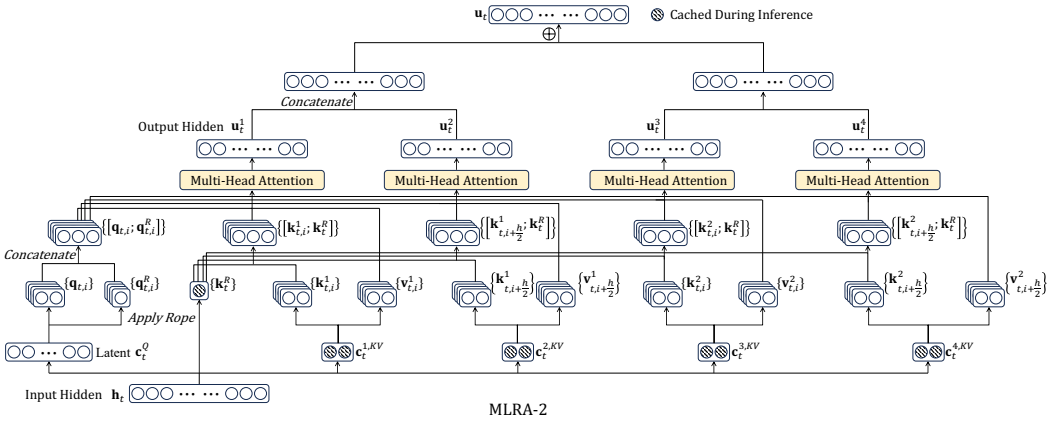

Figure 8: Illustration of MLRA-2.

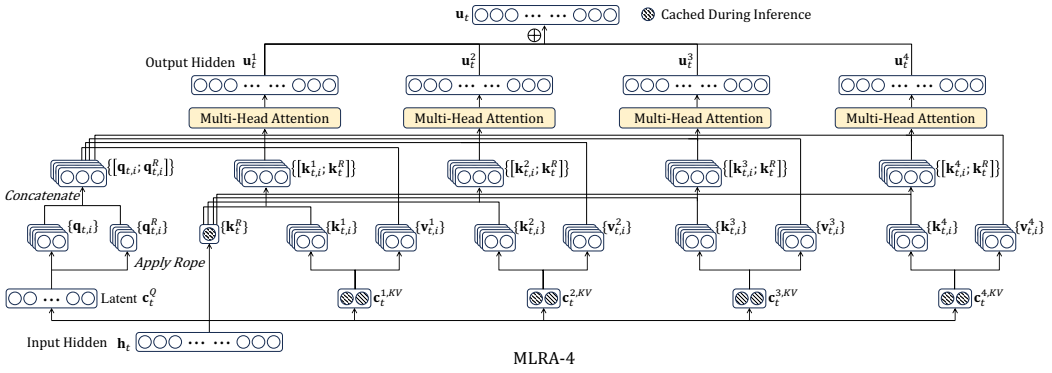

Figure 9: Illustration of MLRA-4.

I RELATED WORK

KV Cache Compression. Recent works (Liu et al., 2023; Anagnostidis et al., 2023; Zhang et al., 2023b; Ge et al., 2024; Xiao et al., 2024; Kim et al., 2024; Zhang et al., 2024b; Nawrot et al.,

2024; Tang et al., 2024; Liu et al., 2024b; Dong et al., 2024; Cai et al., 2024; Liu et al., 2024a; Hooper et al., 2024; Sun et al., 2024; Chen et al., 2024a; Jiang et al., 2024; Li et al., 2024; Xiao et al., 2025; Sun et al., 2025; Meng et al., 2025; Tang et al., 2025) don't introduce new attention mechanisms; instead, they compress the KV cache for pretrained models. Some of these works (Liu et al., 2024b; Hooper et al., 2024) use quantization to store the KV cache in low-bit formats. Some other approaches (Zhang et al., 2023b; Xiao et al., 2024; Li et al., 2024; Xiao et al., 2025) retain important tokens and discard others to compress the KV cache.

Low-Rank Approximation. Low-rank approximation (Hu et al., 2022; Malladi et al., 2023; Zhang et al., 2023a; Dettmers et al., 2023; Lialin et al., 2024; Zhu et al., 2024; Zeng & Lee, 2024; Chen et al., 2024b; Lin et al., 2025; Wang et al., 2025; Chang et al., 2025) are widely used to compress representations to a low-dimensional space, then up-project to recover full representations. These methods greatly reduce trainable parameters (Hu et al., 2022; Dettmers et al., 2023) during fine-tuning and decrease the number of parameters (Lin et al., 2025; Wang et al., 2025) for pretrained models.

System for Attention. FlashAttention (Dao et al., 2022; Dao, 2024; Shah et al., 2024) uses tiling and online softmax to minimize reads and writes between high-bandwidth memory and on-chip SRAM, shifting attention from a memory bottleneck to a compute bottleneck. FlashMLA (Jiashi Li, 2025) avoids explicit KV materialization during attention decoding by absorbing the key up-projection matrices into the queries. The following attention computation is similar to MQA with shared KV states. Inspired by classical virtual memory and paging in operating systems, PagedAttention (Kwon et al., 2023) and vLLM use block-level memory management and preemptive request scheduling to reduce fragmentation and redundant duplication.

Linear Attention. Linear attention (Katharopoulos et al., 2020; Peng et al., 2021; Schlag et al., 2021; Gu et al., 2022; Smith et al., 2023; Sun et al., 2023; Qin et al., 2023; Yang et al., 2024a; Dao & Gu, 2024; Peng et al., 2024; Gu & Dao, 2024; Beck et al., 2024; Zhang et al., 2024a; Yang et al., 2024b; 2025c) reformulates the attention mechanism by substituting the exponential kernel in softmax with a dot product between the query and key vectors. It reduces the memory complexity per decoding step from $\mathcal{O}(n)$ for full attention to $\mathcal{O}(1)$.