Towards Economical Inference: Enabling Deepseek's Multi-Head Latent Attention in Any Transformer-based LLMs

Anonymous ACL submission

Abstract

Multi-head Latent Attention (MLA) is an innovative architecture proposed by DeepSeek, designed to ensure efficient and economical inference by significantly compressing the Key-Value (KV) cache into a latent vector. Compared to MLA, standard LLMs employing Multi-Head Attention (MHA) and its variants such as Grouped-Query Attention (GQA) exhibit significant cost disadvantages. Enabling well-trained LLMs (e.g., Llama) to rapidly adapt to MLA without pre-training from scratch is both meaningful and challenging. This paper proposes the first data-efficient fine-tuning method for transitioning from MHA to MLA (**MHA2MLA**), which includes two key components: for *partial-RoPE*, we remove RoPE from dimensions of queries and keys that contribute less to the attention scores, for *low-rank approximation*, we introduce joint SVD approximations based on the pre-trained parameters of keys and values. These carefully designed strategies enable MHA2MLA to recover performance using only a small fraction (3‰ to 6‰) of the data, significantly reducing inference costs while seamlessly integrating with compression techniques such as KV cache quantization. For example, the KV cache size of Llama2-7B is reduced by 92.19%, with only a 0.5% drop in LongBench performance.[1]

1 Introduction

The rapid advancement of large language models (LLMs) has significantly accelerated progress toward artificial general intelligence (AGI), with model capabilities scaling predictably with parameter counts (Kaplan et al., 2020). However, these gains come at a steep cost: escalating computational demands for training and degraded inference throughput, resulting in substantial energy consumption and carbon emissions (Strubell et al., 2019).

As downstream tasks grow increasingly complex, long-context processing and computationally intensive inference have become central to LLM applications. A key bottleneck lies in the memory footprint of the Key-Value (KV) cache inherent to the Multi-Head Attention (MHA) mechanism (Vaswani et al., 2017), which scales linearly with sequence length and model size. To mitigate this, variants like Grouped-Query Attention (GQA, 2023) and Multi-Query Attention (MQA, 2019) have been explored. However, these methods reduce not only the KV cache size but also the number of parameters in the attention, leading to performance degradation. The DeepSeek introduces Multi-Head Latent Attention (MLA), an attention mechanism equipped with low-rank key-value joint compression. Empirically, MLA achieves superior performance compared with MHA, and meanwhile significantly reduces the KV cache during inference, thus boosting the inference efficiency.

A critical yet unexplored question arises: **Can LLMs originally well-trained for MHA be adapted to enabling MLA for inference?** The inherent architectural disparities between MHA and MLA render zero-shot transfer impractical, while the prohibitive cost of pretraining from scratch makes this transition both technically challenging and underexplored in existing research. To address this gap, we propose the first carefully designed MHA2MLA framework that maximizes parameter reuse from pre-trained MHA networks while aligning the KV cache storage and inference process with MLA's paradigm (Figure 1). Our framework features two pivotal technical innovations: partial rotary position embedding (partial RoPE) and low-rank approximation. The primary objective of MHA2MLA is to achieve data-efficient performance recovery - restoring architecture-induced capability degradation using minimal fine-tuning data.

The inherent incompatibility between MLA's

[1] The source code and models will be publicly accessible.

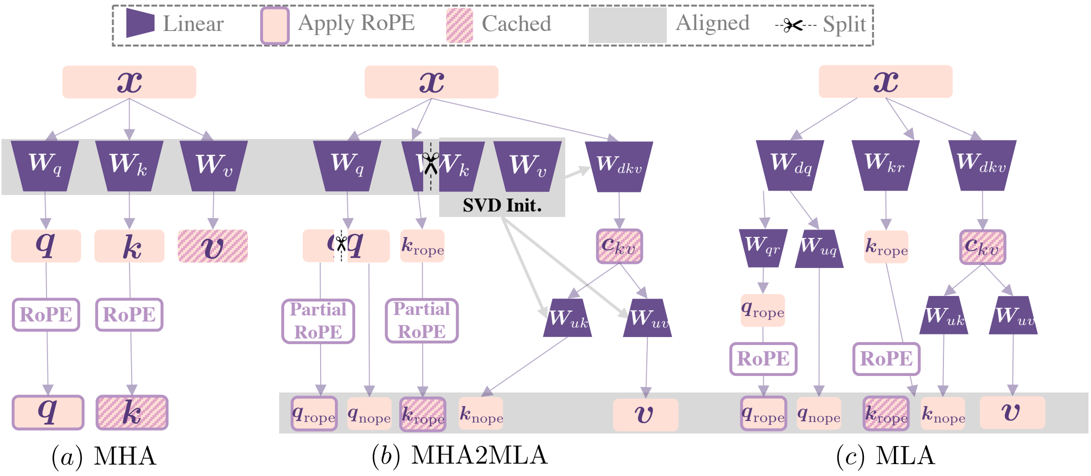

Figure 1: The diagram illustrates the MHA, MLA, and our MHA2MLA. It can be seen that the "cached" part is fully aligned with MLA after MHA2MLA. The input to the attention module is also completely aligned with MLA (the aligned region below). Meanwhile, the parameters in MHA2MLA maximize the use of pre-trained parameters from MHA (the aligned region above).

inference acceleration mechanism and RoPE necessitates architectural compromises. DeepSeek's solution preserves PEs in limited dimensions while compressing others, requiring strategic removal of RoPE dimensions (converting them to NoPE) in MHA to achieve MLA alignment. While higher removal ratios enhance compression efficiency, they exacerbate performance degradation, creating an efficiency-capability trade-off. Through systematically exploring RoPE removal strategies, we identify that contribution-aware dimension selection (retaining top-k dimensions ranked by attention score impact) optimally balances these competing objectives. Although previous studies have investigated training partial-RoPE LLMs from scratch(Black et al., 2021; Barbero et al., 2024), our work pioneers data-efficient fine-tuning for full-to-partial RoPE conversion in LLMs.

MLA reduces memory footprint by projecting keys and values into a low-rank latent representation space (stored in the KV cache). MHA2MLA can also apply low-rank approximation to the values and keys stripped of RoPE (NoPE dimensions). By performing Singular Value Decomposition (SVD) on the pre-trained parameter matrices $\boldsymbol{W}_v$ and $\boldsymbol{W}_k$ corresponding to the NoPE subspaces, we compress these components into a latent space while maximizing the retention of knowledge learned by the original model.

Our main contributions are:

- we introduce MHA2MLA, the first parameter-efficient fine-tuning framework that adapts pre-trained MHA-based LLMs to the MLA architecture using only 3‰ to 6‰ of training data without training from scratch.
- we demonstrate that the MHA2MLA architecture can be integrated with KV-cache quantization to achieve more economical inference (up to a 96.87% reduction).
- we conduct experiments across four model sizes (from 135M to 7B, covering both MHA and GQA), and detailed ablation studies to provide guidance and insights for MHA2MLA.

2 Preliminary

2.1 Multi-Head Attention (MHA)

Given an input sequence $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_l\} \in \mathbb{R}^{l \times d}$, standard MHA (Vaswani et al., 2017) projects each token $\boldsymbol{x}_i$ into queries $\boldsymbol{q}_i^{(h)} = \boldsymbol{x}_i \boldsymbol{W}_q^{(h)}$, keys $\boldsymbol{k}_i^{(h)} = \boldsymbol{x}_i \boldsymbol{W}_k^{(h)}$, and values $\boldsymbol{v}_i^{(h)} = \boldsymbol{x}_i \boldsymbol{W}_v^{(h)}$, where $\boldsymbol{W}_q^{(h)}, \boldsymbol{W}_k^{(h)}, \boldsymbol{W}_v^{(h)} \in \mathbb{R}^{d \times d_h}$ for each head $h \in \{1, \dots, n_h\}$. The Rotary positional encoding (RoPE, 2024) is applied to queries and keys (e.g., $\boldsymbol{q}_{i,\text{rope}}^{(h)} = \text{RoPE}(\boldsymbol{q}_i^{(h)})$), followed by scaled dot-product attention[2]:

$$\boldsymbol{o}_i^{(h)} = \text{Softmax}\left(\boldsymbol{q}_{i,\text{rope}}^{(h)} \boldsymbol{k}_{\le i,\text{rope}}^{(h)\top}\right) \boldsymbol{v}_{\le i}^{(h)},$$

$$\text{MHA}(\boldsymbol{x}_i) = \left[\boldsymbol{o}_i^{(1)}, \dots, \boldsymbol{o}_i^{(n_h)}\right] \boldsymbol{W}_o, \qquad (1)$$

where $\boldsymbol{W}_o \in \mathbb{R}^{(n_h d_h) \times d}$ and $[\cdot, \cdot]$ means vector concatenate. During autoregressive inference, MHA stores the KV cache $\{\boldsymbol{k}_{\text{rope}}^{(h)}, \boldsymbol{v}^{(h)}\}_{h=1}^{n_h}$ of size

[2] We ignore here the $\frac{1}{\sqrt{d}}$ scaling factor for ease of notation.

$\mathcal{O}(2ln_hd_h)$, growing linearly with sequence length l, posing memory bottlenecks.

Variants: Grouped-Query Attention (GQA, 2023) shares keys/values across n_g groups ($n_g \ll n_h$) to reduce the KV cache. For each head h, it maps to group $g = \lfloor h/n_g \rfloor$:

$$\begin{aligned} \boldsymbol{o}_i^{(h)} &= \mathrm{Softmax}\left(\boldsymbol{q}_{i,\text{rope}}^{(h)} \boldsymbol{k}_{\leq i,\text{rope}}^{(g)\top}\right) \boldsymbol{v}_{\leq i}^{(g)}, \\ \mathrm{GQA}(\boldsymbol{x}_i) &= \left[\boldsymbol{o}_i^{(1)}, \ldots, \boldsymbol{o}_i^{(n_h)}\right] \boldsymbol{W}_o. \qquad (2) \end{aligned}$$

Multi-Query Attention (MQA, 2019) is a special case of GQA with $n_g = 1$, i.e., all heads share a single global key/value. While reducing the KV cache to $\mathcal{O}(2ln_gd_h)$, these methods degrade performance due to parameter pruning.

2.2 Multi-Head Latent Attention (MLA)

MLA (DeepSeek-AI et al., 2024) introduces a hybrid architecture that decouples PE from latent KV compression. For each head h, the input $\boldsymbol{x}_i$ is projected into two complementary components:

Position-Aware Component A subset of dimensions retains PE to preserve positional sensitivity:

$$\boldsymbol{q}_{i,\text{rope}}^{(h)}, \boldsymbol{k}_{i,\text{rope}}^{(h)} = \mathrm{RoPE}\left(\boldsymbol{x}_i \boldsymbol{W}_{dq}^{(h)} \boldsymbol{W}_{qr}^{(h)}, \boldsymbol{x}_i \boldsymbol{W}_{kr}^{(h)}\right),$$

where $\boldsymbol{W}_{dq}^{(h)} \in \mathbb{R}^{d \times d_q}$, $\boldsymbol{W}_{qr}^{(h)} \in \mathbb{R}^{d_q \times d_r}$, $\boldsymbol{W}_{kr}^{(h)} \in \mathbb{R}^{d \times d_r}$ project queries/keys into a RoPE-preserved component of dimension d_r.

Position-Agnostic Component The remaining dimensions d_c are stripped of PE (i.e., NoPE), $\boldsymbol{k}_{i,\text{nope}}^{(h)}$ and $\boldsymbol{v}_i^{(h)}$ and compressed into a shared latent vector $\boldsymbol{c}_{i,kv}^{(h)}$:

$$\begin{aligned} \boldsymbol{q}_{i,\text{nope}}^{(h)} &= \boldsymbol{x}_i \boldsymbol{W}_{dq}^{(h)} \boldsymbol{W}_{qc}^{(h)}, \\ \boldsymbol{c}_{i,kv}^{(h)} &= \boldsymbol{x}_i \boldsymbol{W}_{dkv}^{(h)}, \\ \boldsymbol{k}_{i,\text{nope}}^{(h)}, \boldsymbol{v}_i^{(h)} &= \boldsymbol{c}_{i,kv}^{(h)} \boldsymbol{W}_{uk}^{(h)}, \boldsymbol{c}_{i,kv}^{(h)} \boldsymbol{W}_{uv}^{(h)}, \end{aligned}$$

where $\boldsymbol{W}_{qc}^{(h)} \in \mathbb{R}^{d_q \times d_c}$, $\boldsymbol{W}_{dkv}^{(h)} \in \mathbb{R}^{d \times d_{kv}}$, $\boldsymbol{W}_{uk}^{(h)} \in \mathbb{R}^{d_{kv} \times d_c}$, $\boldsymbol{W}_{uv}^{(h)} \in \mathbb{R}^{d_{kv} \times d_h}$. Note that $d_r + d_c = d_h$. The attention output of MLA combines both components:

$$\begin{aligned} \boldsymbol{o}_i^{(h)} &= \mathrm{Softmax}\left(\boldsymbol{q}_{i,\text{rope}}^{(h)} \boldsymbol{k}_{\leq i,\text{rope}}^{(h)\top} + \boldsymbol{q}_{i,\text{nope}}^{(h)} \boldsymbol{k}_{\leq i,\text{nope}}^{(h)\top}\right) \\ &\quad \cdot \boldsymbol{v}_{\leq i}^{(h)} \\ \mathrm{MLA}(\boldsymbol{x}_i) &= \left[\boldsymbol{o}_i^{(1)}, \ldots, \boldsymbol{o}_i^{(n)}\right] \cdot \boldsymbol{W}_o. \qquad (3) \end{aligned}$$

Unlike MHA and its variants, MLA stores the latent vector $\boldsymbol{c}_{kv}^{(h)}$ and $\boldsymbol{k}_{i,\text{rope}}^{(h)}$ ($\mathcal{O}\left(ln_h(d_r + d_{kv})\right)$) instead of full-rank $\boldsymbol{k}, \boldsymbol{v}$ ($\mathcal{O}(2ln_hd_h)$), where $(d_r + d_{kv}) \ll d_h$.

Why does MLA need to separate RoPE and NoPE? MLA introduces matrix merging techniques for the NoPE portion during inference, effectively reducing memory usage. For the dot product operation $\boldsymbol{q}_{i,\text{nope}}^{(h)} \boldsymbol{k}_{j,\text{nope}}^{(h)\top}$, the following identity transformation can be applied:

$$\begin{aligned} \boldsymbol{q}_{i,\text{nope}} \boldsymbol{k}_{j,\text{nope}}^{\top} &= \left(\boldsymbol{x}_i \boldsymbol{W}_{dq} \boldsymbol{W}_{qc}\right) \left(\boldsymbol{c}_{j,kv} \boldsymbol{W}_{uk}\right)^{\top} \\ &= \boldsymbol{x}_i \left(\boldsymbol{W}_{dq} \boldsymbol{W}_{qc} \boldsymbol{W}_{uk}^{\top}\right) \boldsymbol{c}_{j,kv}^{\top} \end{aligned}$$

where $\left(\boldsymbol{W}_{dq} \boldsymbol{W}_{qc} \boldsymbol{W}_{uk}^{\top}\right)$ can be pre-merged into a single matrix, and $\boldsymbol{c}_{j,kv}$ is already stored in the KV cache. As for the RoPE portion, the RoPE$(\cdot)$ function multiplies the input vector by the rotation matrix (e.g., RoPE($\boldsymbol{q}_i$) = $\boldsymbol{q}_i \boldsymbol{R}_i$, $\boldsymbol{R}_i$'s specific form will be introduced in Section 3.1). Therefore, the identity transformation becomes as follows:

$$\begin{aligned} \boldsymbol{q}_{i,\text{rope}} \boldsymbol{k}_{j,\text{rope}}^{\top} &= \left(\boldsymbol{x}_i \boldsymbol{W}_{dq} \boldsymbol{W}_{qr} \boldsymbol{R}_i\right) \left(\boldsymbol{x}_j \boldsymbol{W}_{kr} \boldsymbol{R}_j\right)^{\top} \\ &= \boldsymbol{x}_i \left(\boldsymbol{W}_{dq} \boldsymbol{W}_{qc} \boldsymbol{R}_{j-i} \boldsymbol{W}_{kr}^{\top}\right) \boldsymbol{x}_j^{\top} \end{aligned}$$

Since $\left(\boldsymbol{W}_{dq} \boldsymbol{W}_{qc} \boldsymbol{R}_{j-i} \boldsymbol{W}_{kr}^{\top}\right)$ is related to the relative position $j - i$, it cannot be merged into a fixed matrix. Considering that the relative distances in LLMs can be very long, such as 128K, the RoPE portion is better suited to be computed using the original form.

3 MHA2MLA

3.1 Partial-RoPE

To enable migration from standard MHA to MLA, we propose partial-RoPE finetuning, a strategy that removes RoPE from a targeted proportion of dimensions and converts them into NoPE. Critically, while prior work has explored training LLMs with partial-RoPE from scratch (achieving marginally better perplexity than full-RoPE (Black et al., 2021; Barbero et al., 2024)), no existing method addresses how to efficiently adapt pre-trained full-RoPE models (e.g., Llama) to partial-RoPE without costly retraining. Our work bridges this gap by systematically evaluating partial-RoPE variants to identify the most data-efficient fine-tuning protocol for recovering model performance post-adaptation.

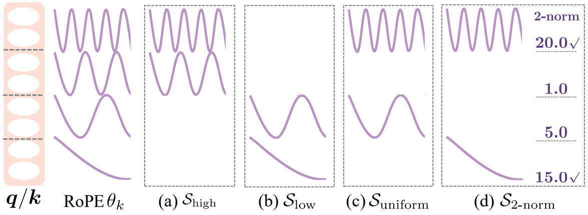
2-norm
20.0✓
1.0
5.0
15.0✓
q/k
RoPE θ_k
(a) $\mathcal{S}_{\text{high}}$
(b) $\mathcal{S}_{\text{low}}$
(c) $\mathcal{S}_{\text{uniform}}$
(d) $\mathcal{S}_{\text{2-norm}}$

Figure 2: Illustration of $\mathcal{S}_{\text{high}}$, $\mathcal{S}_{\text{low}}$, $\mathcal{S}_{\text{uniform}}$, $\mathcal{S}_{\text{2-norm}}$. Where $d_h = 8$ and $r = 2$.

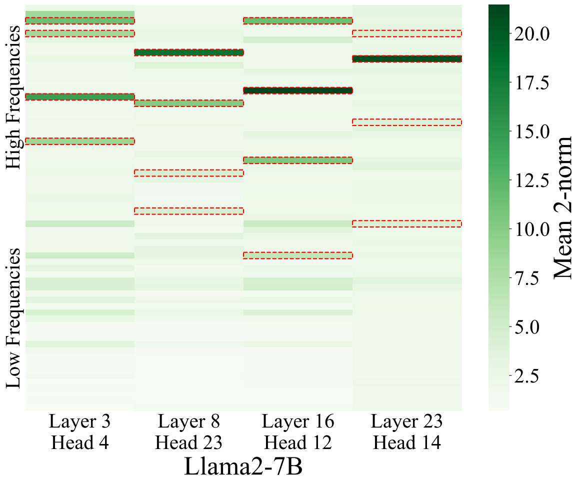

Figure 3: Visualization of Head-wise 2-norm Contribution for Llama2-7B. We randomly selected 4 heads, and the red dashed box highlights the top-4 frequency subspaces chosen when $r = 4$. It can be seen that different heads tend to focus on different frequency subspaces, which validates the rationality of our $\mathcal{S}_{\text{2-norm}}$ method.

MHA's Full-RoPE encodes positional information into queries and keys through frequency-specific rotations. Formally, given a query vector $\boldsymbol{q}_i \in \mathbb{R}^{d_h}$ and key vector $\boldsymbol{k}_i \in \mathbb{R}^{d_h}$, we partition them into 2D chunks:

$$\boldsymbol{q}_i, \boldsymbol{k}_i = \left[\boldsymbol{q}_i^{[2k,2k+1]}\right]_{0 \leq k < \frac{d_h}{2}}, \left[\boldsymbol{k}_i^{[2k,2k+1]}\right]_{0 \leq k < \frac{d_h}{2}},$$

where $\boldsymbol{q}_i^{[2k,2k+1]} \in \mathbb{R}^2$ denotes the k-th 2D subspace. Each chunk undergoes a rotation by position-dependent angles $\theta_k = \beta^{-2k/d_h}$, forming a spectrum of wavelengths. High-frequency components, e.g., $k = 0$, rotate rapidly at 1 radian per token. Low-frequency components, e.g., $k = \frac{d_h}{2} - 1$, rotate slowly at $\sim \beta^{1/d_h}$ radians per token. The base wavelength β, typically set to 10^4 (Su et al., 2024) or 5×10^5.

Formally, for each 2D chunk $\boldsymbol{q}_i^{[2k,2k+1]}$ and $\boldsymbol{k}_i^{[2k,2k+1]}$, the rotation matrix at position i is defined as:

$$\boldsymbol{R}_i^{[2k,2k+1]}(\theta_k) = \begin{bmatrix} \cos(i\theta_k) & -\sin(i\theta_k) \\ \sin(i\theta_k) & \cos(i\theta_k) \end{bmatrix}.$$

Thus, applying RoPE to queries and keys becomes:

$$\boldsymbol{q}_{i,rope} = \left[\boldsymbol{R}_i^{[2k,2k+1]}(\theta_k)\boldsymbol{q}_i^{[2k,2k+1]}\right]_{0 \leq k < \frac{d_h}{2}},$$

$$\boldsymbol{k}_{i,rope} = \left[\boldsymbol{R}_i^{[2k,2k+1]}(\theta_k)\boldsymbol{k}_i^{[2k,2k+1]}\right]_{0 \leq k < \frac{d_h}{2}}.$$

Full-RoPE to Partial-RoPE Strategies Given r retained rotational subspaces($r \ll$ total subspaces $\frac{d_h}{2}$, we propose four strategies (Illustrated in Figure 2) to select which r subspaces preserve RoPE encoding:

High-Frequency Preservation retain the r fastest-rotating (high-frequency) subspaces:

$$\mathcal{S}_{\text{high}} = \{k \,|\, 0 \leq k < r\}.$$

It is consistent with the p-RoPE method proposed in Barbero et al. (2024), where they explored settings in which r constituted 25%, 50%, and 75% of the total subspaces, and observed a slight advantage over full-RoPE in LLMs trained from scratch.

Low-Frequency Preservation retain the r slowest-rotating (low-frequency) subspaces:

$$\mathcal{S}_{\text{low}} = \left\{k \,\middle|\, \frac{d_h}{2} - r \leq k < \frac{d_h}{2}\right\}.$$

It was chosen as a controlled experiment for the high-frequency strategy.

Uniform Sampling select r subspaces with equidistant intervals:

$$\mathcal{S}_{\text{uniform}} = \left\{\left\lfloor k\frac{d_h}{2r}\right\rfloor \,\middle|\, 0 \leq k < r\right\}$$

This balances high- and low-frequency components through geometric spacing. In practice, $2r$ typically divides d_h. It is similar to the partial RoPE used in GPT-Neo (Black et al., 2021).

Head-wise 2-norm Contribution Barbero et al. (2024) were the first to propose the 2-norm contribution to investigate whether these frequencies are utilized and how they are helpful. This approach is based on the observation that, according to the Cauchy-Schwarz inequality, the influence of the k-th frequency subspace on the attention logits is upper-bounded by the 2-norm of the corresponding query and key components, i.e., $\left|\left\langle \mathbf{q}_i^{[2k,2k+1]}, \mathbf{k}_j^{[2k,2k+1]} \right\rangle\right| \leqslant \left\|\mathbf{q}_i^{[2k,2k+1]}\right\| \left\|\mathbf{k}_j^{[2k,2k+1]}\right\|$. For each head h, we compute the mean 2-norm score for each subspace

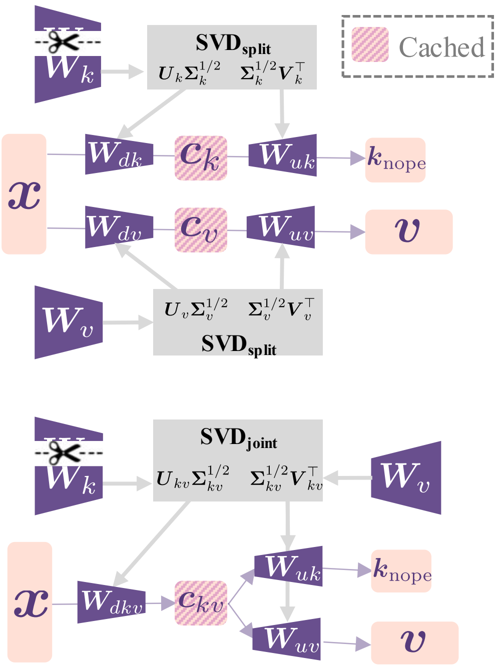

Figure 4: Illustration of **SVD$_{\text{split}}$** and **SVD$_{\text{joint}}$**.

in an LLM over long sequences [3]. Then, we propose to rank all subspaces by their 2-norm score and select the top-r:

$$\mathcal{S}_{\text{2-norm}} = \underset{0\le k<\frac{d_h}{2}}{\text{top-}r}\left(\left\|\mathbf{q}_*^{[2k,2k+1]}\right\|\left\|\mathbf{k}_*^{[2k,2k+1]}\right\|\right).$$

This head-specific selection adaptively preserves rotation-critical subspaces. Figure 3 visualizes the 2-norm of Llama2-7B's four heads.

We will analyze the effectiveness of the four strategies and conduct an ablation study on the essential hyperparameter r in Section 4.3. For all strategies, non-selected subspaces ($k \notin \mathcal{S}$) become NoPE dimensions, enabling seamless integration with MLA's latent compression.

3.2 Low-rank Approximation

After transitioning from full RoPE to partial RoPE, we obtain the first component of the KV cache in MLA, represented as: $\boldsymbol{k}_{i,rope} = \left[\boldsymbol{R}_i^{[2k,2k+1]}(\theta_k)\boldsymbol{k}_i^{[2k,2k+1]}\right]_{k\in\mathcal{S}}$. Our next objective is to derive the second component, $\boldsymbol{c}_{i,kv} \in \mathbb{R}^{d_{kv}}$, which serves as a low-rank representation of $\boldsymbol{k}_{i,\text{nope}}$ and $\boldsymbol{v}_i$.

Given the keys $\boldsymbol{k}_i = \boldsymbol{x}_i\boldsymbol{W}_k$ and values $\boldsymbol{v}_i = \boldsymbol{x}_i\boldsymbol{W}_v$ in MHA, we first extract the subspace of $\boldsymbol{W}_k$ corresponding to $\boldsymbol{k}_{i,\text{nope}}$, i.e., the dimensions not included in $\mathcal{S}$, yielding: $\boldsymbol{k}_{i,\text{nope}} = \boldsymbol{x}_i\boldsymbol{W}_{k,\text{nope}}$. We propose two Singular Value Decomposition (SVD)-based strategies (Illustrated in Figure 4) to preserve pre-trained knowledge while achieving rank reduction:

Decoupled SVD (SVD$_{\text{split}}$) Separately decompose $\boldsymbol{W}_{k,\text{nope}}$ and $\boldsymbol{W}_v$ into truncated SVDs, allocating $d_{kv}/2$ dimensions to each:

$$\boldsymbol{W}_{k,\text{nope}} = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^\top, \quad \boldsymbol{W}_v = \boldsymbol{U}_v\boldsymbol{\Sigma}_v\boldsymbol{V}_v^\top,$$

where $\boldsymbol{U}_k, \boldsymbol{U}_v, \boldsymbol{V}_k, \boldsymbol{V}_v \in \mathbb{R}^{d\times\frac{d_{kv}}{2}}$, $\boldsymbol{\Sigma}_k, \boldsymbol{\Sigma}_v \in \mathbb{R}^{\frac{d_{kv}}{2}\times\frac{d_{kv}}{2}}$. The down-projection matrices $\boldsymbol{W}_{d*}$ and up-projection matrices $\boldsymbol{W}_{u*}$ become:

$$\boldsymbol{W}_{dk} = \boldsymbol{U}_k\boldsymbol{\Sigma}_k^{1/2}, \quad \boldsymbol{W}_{uk} = \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{V}_k^\top,$$
$$\boldsymbol{W}_{dv} = \boldsymbol{U}_v\boldsymbol{\Sigma}_v^{1/2}, \quad \boldsymbol{W}_{uv} = \boldsymbol{\Sigma}_v^{1/2}\boldsymbol{V}_v^\top.$$

The low-rank representation $\boldsymbol{c}_{i,kv}$ can be constructed using $\boldsymbol{c}_{i,kv} = [\boldsymbol{x}_i\boldsymbol{W}_{dk}, \boldsymbol{x}_i\boldsymbol{W}_{dv}]$.

Joint SVD (SVD$_{\text{joint}}$) To preserve interactions between $\boldsymbol{K}_{\text{nope}}$ and $\boldsymbol{V}$, we jointly factorize the concatenated matrix:

$$[\boldsymbol{W}_{k,\text{nope}}, \boldsymbol{W}_v] = \boldsymbol{U}_{kv}\boldsymbol{\Sigma}_{kv}\boldsymbol{V}_{kv}^\top,$$

where $\boldsymbol{U}_{kv}, \boldsymbol{V}_{kv} \in \mathbb{R}^{d\times d_{kv}}$, $\boldsymbol{\Sigma}_{kv} \in \mathbb{R}^{d_{kv}\times d_{kv}}$. The latent projection is then:

$$\boldsymbol{W}_{dkv} = \boldsymbol{U}_{kv}\boldsymbol{\Sigma}_{kv}^{1/2},$$
$$\boldsymbol{W}_{uk} = \boldsymbol{\Sigma}_{kv}^{1/2}\boldsymbol{V}_{kv}[:, : -d_v], \boldsymbol{W}_{uv} = \boldsymbol{\Sigma}_{kv}^{1/2}\boldsymbol{V}_{kv}[:, d_v :].$$

This jointly optimizes the latent space for both keys and values, i.e., $\boldsymbol{c}_{i,kv} = \boldsymbol{x}_i\boldsymbol{W}_{dkv}$, retaining cross-parameter dependencies critical for autoregressive generation. Section 4.3 shows SVD$_{\text{joint}}$ outperforming SVD$_{\text{split}}$, validating that joint factorization better preserves pre-trained knowledge.

4 Experiment

We evaluate our method on LLMs of varying scales (SmolLM-135M, SmolLM-360M, SmolLM-1B7, Llama2-7B) pre-trained with MHA or GQA. We chose the SmolLM-series[4] because its pretraining data and framework are both open-source, which can minimize the gap in fine-tuning data and processes. We chose Llama2-7B[5] because it is one of the widely used open-source LLMs (although its pretraining data is not open-source, there is a potential gap in fine-tuning data).

[3] The 2-norm calculation detail is placed in Appendix A.

[4] https://huggingface.co/collections/HuggingFaceTB/smollm-6695016cad7167254ce15966

[5] https://huggingface.co/meta-llama/Llama-2-7b

Model		Tokens	KV Mem.	Avg.	MMLU	ARC	PIQA	HS	OBQA	WG
$135\text{M}_{\text{SmolLM}}$		600B		44.50	29.80	42.43	68.06	41.09	33.60	52.01
- *GQA*	$d_{kv}=128$			44.25	29.82	42.05	68.34	41.03	33.20	51.07
	$d_{kv}=32$	2.25B	-68.75%	43.06 $_{-1.19}$	29.50	40.48	66.59	37.99	33.80	49.96
- *GQA2MLA*	$d_{kv}=16$	(3.8‰)	-81.25%	41.84 $_{-2.41}$	28.66	39.95	65.02	36.04	31.60	49.80
	$d_{kv}=8$		-87.50%	40.97 $_{-3.28}$	28.37	38.04	64.69	33.58	30.80	50.36
$360\text{M}_{\text{SmolLM}}$		600B		49.60	33.70	49.82	71.87	51.65	37.60	52.96
- *GQA*	$d_{kv}=128$			49.63	34.01	50.02	71.33	51.43	38.20	52.80
	$d_{kv}=32$	2.25B	-68.75%	47.91 $_{-1.72}$	32.94	48.36	70.73	48.16	36.00	51.30
- *GQA2MLA*	$d_{kv}=16$	(3.8‰)	-81.25%	46.94 $_{-2.69}$	31.55	45.73	70.51	45.80	36.60	51.46
	$d_{kv}=8$		-87.50%	45.04 $_{-4.59}$	30.54	43.33	68.50	42.83	35.00	50.04
$1\text{B7}_{\text{SmolLM}}$		1T		55.90	39.27	59.87	75.73	62.93	42.80	54.85
- *MHA*	$d_{kv}=128$			55.93	39.11	59.19	75.95	62.92	43.40	55.09
	$d_{kv}=32$	6B	-68.75%	54.76 $_{-1.17}$	38.11	57.13	76.12	61.35	42.00	53.83
- *MHA2MLA*	$d_{kv}=16$	(6.0‰)	-81.25%	54.65 $_{-1.28}$	37.87	56.81	75.84	60.41	42.60	54.38
	$d_{kv}=8$		-87.50%	53.61 $_{-2.32}$	37.17	55.50	74.86	58.55	41.20	54.38
$7\text{B}_{\text{Llama2}}$		2T		59.85	41.43	59.24	78.40	73.29	41.80	64.96
- *MHA*	$d_{kv}=256$			60.22	41.63	60.89	77.80	71.98	45.00	63.38
	$d_{kv}=64$	6B	-68.75%	59.51 $_{-0.71}$	41.36	59.51	77.37	71.72	44.20	62.90
- *MHA2MLA*	$d_{kv}=32$	(3.0‰)	-81.25%	59.61 $_{-0.61}$	40.86	59.74	77.75	70.75	45.60	62.98
	$d_{kv}=16$		-87.50%	58.96 $_{-1.26}$	40.39	59.29	77.75	69.70	43.40	63.22

Table 1: Commonsense reasoning ability of four LLMs with MHA2MLA or GQA2MLA. The six benchmarks include MMLU (2021), ARC easy and challenge (ARC, 2018), PIQA (2020), HellaSwag (HS, 2019), OpenBookQA (OBQA, 2018), Winogrande (WG, 2021).

We denote the architectural migration using MHA2MLA and GQA2MLA, respectively.[6] Both MHA2MLA and GQA2MLA adopt *data-efficient full-parameter fine-tuning*, with the head-wise 2-norm selection ($\mathcal{S}_{\text{2-norm}}$) for Partial-RoPE and joint SVD factorization ($\textbf{SVD}_{\textbf{joint}}$) for low-rank approximation as default configurations. Our experiments address three critical questions:

1. How does MHA2MLA minimize accuracy degradation induced by architectural shifts?
2. What does MHA2MLA achieve in the KV cache reduction ratio?
3. Can MHA2MLA integrate with KV cache quantization for compound gains?

4.1 General Tasks

Main Results As shown in Table 1, our method achieves efficient architectural migration across four model scales (135M to 7B) under varying KV cache compression ratios (via latent dimension d_{kv}). First, when comparing the performance of our fine-tuning approach with the original LLM, we observe only minor changes in performance across the four base models: a -0.25% decrease on the 135M, +0.03% on the 360M, +0.03% on the 1B7, and +0.37% on the 7B. This suggests that the fine-tuning data does not significantly degrade or improve the performance of the original model, providing an appropriate experimental setting for the MHA2MLA framework.

Next, as d_{kv} decreases (e.g., from 32 to 16 to 8), the KV cache reduction increases (i.e., from -68.75% to -81.25% to -87.5%), but the performance loss becomes more challenging to recover through fine-tuning. Figure 5 shows the fine-tuning loss curves of 135M (representing GQA) and 7B (representing MHA) under different compression ratios. As the compression ratio increases, the loss difference from the baseline becomes larger. Additionally, we observe that the fluctuation trends of the loss curves are *almost identical*, indicating that our architecture migration does not significantly harm the model's internal knowledge.

We also find that larger models experience less performance degradation when transitioning to the MLA architecture. For example, with compression down to 18.75%, the performance drops by 2.41% for 135M, 1.97% for 360M, 1.28% for 1B7, and 0.61% for 7B, revealing the **potential scaling law of MHA2MLA**. Finally, from the 135M model to the 7B model, the number of tokens required for fine-tuning is only about 0.3% to 0.6% of the pre-training tokens, demonstrating the data efficiency of our method.

Overall, whether using GQA2MLA or MHA2MLA, the architecture transition is achieved with minimal cost, resulting in efficient and economical inference.

[6] The details of the fine-tuning process (including data and hyperparameters) are provided in Appendix B.

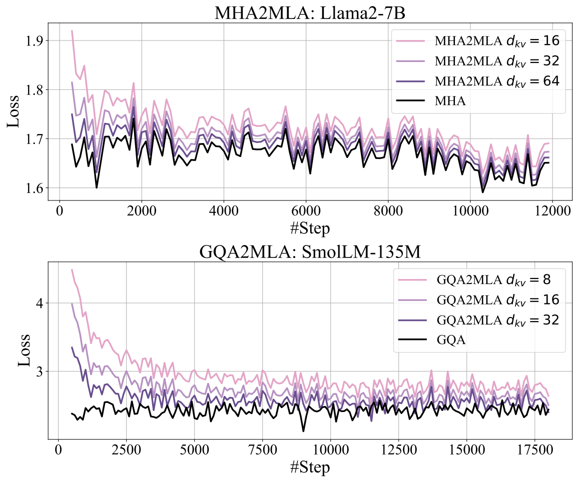

Figure 5: The fine-tuning loss curves under different KV cache storage ratios (with colors ranging from light to dark representing 12.5%, 18.75%, 31.25%, and 100%).

4.2 Long Context Tasks

Settings To evaluate the generative capabilities of the model, we adopt LongBench (Bai et al., 2024) as the benchmark for generation performance. All models are tested using a greedy decoding strategy. The context window size is determined based on the sequence length used during model fine-tuning. We use HQQ (Badri and Shaji, 2023) and Quanto[7] to set caches with different levels of precision to evaluate the performance of the original model as the baseline. Since our method is compatible with KV cache quantization, we also conduct additional experiments to assess the combined effect of both approaches.

Main Results As evidenced in Table 2, MHA2MLA achieves competitive or superior efficiency-accuracy profiles compared to post-training quantization methods on LongBench. While 4-bit quantization incurs modest degradation (-0.2% to -0.4%) at comparable compression ratios, aggressive 2-bit quantization suffers severe performance collapse (-6.2% to -9%) despite 87.5% KV cache reduction. In contrast, MHA2MLA alone attains 87.5% compression (at $d_{kv} = 16$) with only 3% accuracy loss, and further synergizes with 4-bit quantization to reach 92.19%/96.87% compression ($d_{kv} = 64/16$+Int4$_{\text{HQQ}}$) while limiting degradation to -0.5%/-3.2%, outperforming all 2-bit baselines. This highlights that MHA2MLA's latent space design remains orthogonal to numerical precision reduction, enabling **compound efficiency gains** without destructive interference.

Model	Precision	KV Mem.	Avg@LB
7B$_{\text{Llama2}}$	BF16	100.0%	27.4
	Int4$_{\text{HQQ}}$	-75.00%	27.5
	Int4$_{\text{Quanto}}$		27.3
	Int2$_{\text{HQQ}}$	-87.50%	21.2
	Int2$_{\text{Quanto}}$		18.5
$d_{kv} = 64$	BF16	-68.75%	27.1
	Int4$_{\text{HQQ}}$	-92.19%	**26.9**
	Int4$_{\text{Quanto}}$		**26.8**
$d_{kv} = 32$	BF16	-81.25%	26.3
	Int4$_{\text{HQQ}}$	-95.31%	**26.1**
	Int4$_{\text{Quanto}}$		**26.1**
$d_{kv} = 16$	BF16	-87.50%	**24.4**
	Int4$_{\text{HQQ}}$	-96.87%	**24.2**
	Int4$_{\text{quanto}}$		**23.4**

Table 2: Evaluation results of Llama2-7B and MHA2MLA on LongBench. **Bold** indicates compression ratios greater than or equal to Int2 quantization while also achieving performance higher than Int2 quantization.

4.3 Ablation Study

Four Partial-RoPE strategies: $\mathcal{S}_{\text{high}}$, $\mathcal{S}_{\text{low}}$, $\mathcal{S}_{\text{uniform}}$, $\mathcal{S}_{\text{2-norm}}$ Table 3 presents the results of four strategies for converting full-RoPE to partial-RoPE. First, when comparing the four strategies with full-RoPE, we observed that the low-frequency retention strategy, $\mathcal{S}_{\text{low}}$, incurred the greatest performance loss (a reduction of -6.49%@135M and -1.21%@1B7), whereas the high-frequency retention strategy, $\mathcal{S}_{\text{high}}$, experienced significantly less degradation (a reduction of -0.85%@135M and -0.76%@1B7), underscoring the importance of high-frequency subspaces. Both $\mathcal{S}_{\text{uniform}}$ and $\mathcal{S}_{\text{2-norm}}$ yielded better performance, the $\mathcal{S}_{\text{uniform}}$ preserves subspaces across the frequency spectrum, while the $\mathcal{S}_{\text{2-norm}}$ retains subspaces based on their contribution to the attention scores. We choose $\mathcal{S}_{\text{2-norm}}$ as the default configuration because the removed subspaces (i.e., NoPE) are more suitable for the (SVD-based) low-rank approximation.

Two SVD-based low-rank approximations: SVD$_{\text{split}}$, SVD$_{\text{joint}}$ The last two rows of each group in Table 3 compare the effects of the two SVD methods. We observe that, on both LLMs, the SVD$_{\text{joint}}$ method consistently outperforms SVD$_{\text{split}}$, yielding an average performance improvement of 0.92% on the 135M model and 0.74% on the 1B7

[7] https://huggingface.co/blog/quanto-introduction

Model	**Tokens**	**Avg@CS**
$135\text{M}_{\text{SmolLM}}$	600B	44.50
- *full-rope*	2.25B	44.25
- $\mathcal{S}_{\text{high}}$		43.40 $_{-0.85}$
- $\mathcal{S}_{\text{low}}$		37.76 $_{-6.49}$
- $\mathcal{S}_{\text{uniform}}$		43.76 $_{-0.49}$
- $\mathcal{S}_{\text{2-norm}}$		**43.77** $_{-0.48}$
- $\mathcal{S}_{\text{high}}$ + $\text{SVD}_{\text{joint}}$	2.25B	41.04 $_{-3.21}$
- $\mathcal{S}_{\text{uniform}}$ + $\text{SVD}_{\text{joint}}$		41.77 $_{-2.48}$
- $\mathcal{S}_{\text{2-norm}}$ + $\text{SVD}_{\text{joint}}$		**41.84** $_{-2.41}$
- $\mathcal{S}_{\text{2-norm}}$ + $\text{SVD}_{\text{split}}$		40.92 $_{-3.33}$
$1\text{B}7_{\text{SmolLM}}$	1T	55.90
- *full-rope*	6B	55.93
- $\mathcal{S}_{\text{high}}$		55.17 $_{-0.76}$
- $\mathcal{S}_{\text{low}}$		54.72 $_{-1.21}$
- $\mathcal{S}_{\text{uniform}}$		**55.31** $_{-0.62}$
- $\mathcal{S}_{\text{2-norm}}$		55.10 $_{-0.83}$
- $\mathcal{S}_{\text{high}}$ + $\text{SVD}_{\text{joint}}$	6B	54.41 $_{-1.52}$
- $\mathcal{S}_{\text{uniform}}$ + $\text{SVD}_{\text{joint}}$		54.30 $_{-1.63}$
- $\mathcal{S}_{\text{2-norm}}$ + $\text{SVD}_{\text{joint}}$		**54.65** $_{-1.28}$
- $\mathcal{S}_{\text{2-norm}}$ + $\text{SVD}_{\text{split}}$		53.91 $_{-2.02}$

Table 3: Reasoning ability of ablation studies.

model. It indicates that $\text{SVD}_{\text{joint}}$ emerges as the clear default choice.

5 Related Work

Efficient Attention Architectures The standard Multi-Head Attention (MHA, 2017) mechanism's quadratic complexity in context length has spurred numerous efficiency innovations. While MHA remains foundational, variants like Multi-Query Attention (MQA) and Grouped-Query Attention (GQA, 2023) reduce memory overhead by sharing keys/values across heads—albeit at the cost of parameter pruning and performance degradation. Parallel efforts, such as Linear Transformers (Guo et al., 2019; Katharopoulos et al., 2020; Choromanski et al., 2021), RWKV (Peng et al., 2023), and Mamba (Gu and Dao, 2023), replace softmax attention with linear recurrences or state-space models, but struggle to match the expressiveness of standard attention in autoregressive generation.

Multi-Head Latent Attention (MLA) (DeepSeek-AI et al., 2024) distinguishes itself by compressing KV caches into low-rank latent vectors without pruning attention parameters. Our work bridges MLA with mainstream architectures (MHA/GQA), enabling seamless migration via data-efficient fine-tuning. Notably, while many linear attention variants abandon softmax query-key interactions (e.g., through kernel approximations), architectures preserving a query-key dot product structure—even in factorized forms—remain compatible with our MHA2MLA framework.

Economical Key-Value Cache The memory footprint of KV caches has become a critical bottleneck for long-context inference. Recent advances fall into three categories:

Innovative Architecture methods like MLA (DeepSeek-AI et al., 2024), MiniCache (Liu et al., 2024a), and MLKV (Zuhri et al., 2024) share or compress KV representations across layers or heads. While effective, cross-layer sharing risks conflating distinct attention patterns, potentially harming task-specific performance. Only MLA has been successfully validated in Deepseek's LLMs.

Quantization techniques such as GPTQ (Frantar et al., 2022), FlexGen (Sheng et al., 2023), and KIVI (Liu et al., 2024b) store KV caches in low-bit formats (e.g., 2-bit), achieving memory savings with precision loss.

Dynamic Pruning approaches like A2SF (Jo and Shin, 2024) and SnapKV (Li et al., 2024) prune "less important" tokens from the KV cache. However, token pruning risks discarding critical long-range dependencies, while head pruning (e.g., SliceGPT (Ashkboos et al., 2024), Sheared (Xia et al., 2024), and Simple Pruning (Sun et al., 2024)) irreversibly reduces model capacity.

Our MHA2MLA method achieves the migration of standard Transformer-based LLMs to the more economical MLA architecture and has demonstrated its ability to integrate with KV quantization techniques to realize a ~97% cache saving. It is also theoretically compatible with other methods like pruning.

6 Conclusion

This work addresses the critical challenge of adapting pre-trained MHA-based LLMs (or variants) to the KV-cache-efficient MLA architecture. By introducing MHA2MLA with contribution-aware partial-RoPE removal and SVD-driven low-rank projection, we achieve near-lossless compression of KV cache (up to 96.87% size reduction for Llama2-7B) while requiring only 3‰ to 6‰of training data. The framework demonstrates strong compatibility with existing compression techniques and maintains commonsense reasoning and long-context processing capabilities, offering a practical pathway for deploying resource-efficient LLMs without sacrificing performance. Our results underscore the feasibility of architectural migration for LLMs through targeted parameter reuse and data-efficient fine-tuning.

Limitations

Verification on More LLMs Considering that MHA2MLA can significantly reduce inference costs, it is worthwhile to validate it on larger and more diverse open-source LLMs. However, constrained by our computation resources, models like Llama3 require fine-tuning on a 128K context length to mitigate performance degradation from continued training, so we did not perform such experiments. Furthermore, since Deepseek has not yet open-sourced the tensor-parallel inference framework for MLA, it is currently challenging to explore models larger than 7B. This will be addressed in our future work.

Parameter-Efficient MHA2MLA Fine-tuning This paper primarily focuses on the data efficiency of MHA2MLA. Since the architectural transformation does not involve the Feed-Forward (FFN) module, future work could explore parameter-efficient MHA2MLA fine-tuning, for example by freezing the FFN module and/or freezing the parameters in the queries and keys that correspond to the retained RoPE. This could further reduce the cost of the MHA2MLA transition.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Velickovic. 2024. Round and round we go! what makes rotary positional encodings useful? *CoRR*, abs/2410.06205.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, Tao Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, and Xiaowen Sun. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1315–1325. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hyun-rae Jo and Dongkun Shin. 2024. A2SF: accumulative attention scoring with forgetting factor for token pruning in transformer decoder. *CoRR*, abs/2407.20485.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Reza Haffari, and Bohan Zhuang. 2024a. Minicache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanislaw Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14048–14077. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single GPU. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31094–31116. PMLR.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

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

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International*

Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Zayd Muhammad Kawakibi Zuhri, Muhammad Farid Adilazuarda, Ayu Purwarianti, and Alham Fikri Aji. 2024. MLKV: multi-layer key-value heads for memory efficient transformer decoding. *CoRR*, abs/2406.09297.

A The Calculation of 2-norm Score

To compute the 2-norm scores for each attention head, we selected 1,024 samples from the training dataset. The proportions of the subsets and sequence length used during the 2-norm computation are consistent with those used during fine-tuning. First, we calculate the query vectors and key vectors for each head. Then, for each rotational subspace of the vectors, we compute the 2-norm scores. Finally, the 2-norm scores of the query and key vectors are aggregated within each subspace. If the model employs Grouped-Query Attention (GQA), the 2-norm scores are averaged within each GQA group, and the scores are shared between the groups.

B The Details of Fine-tuning

Data We fine-tune our model using the pretraining corpus from SmolLm[8]. The dataset consists of fineweb-edu-dedup, cosmopedia-v2, python-edu, open-web-math, and StackOverflow. The first three datasets are part of the smollm-corpus[9] curated by HuggingFaceTB. Fineweb-edu-dedup is a high-quality dataset filtered by HuggingFaceTB from education-related webpages. Similarly, HuggingFaceTB filtered Python code snippets from The Stack to construct the python-edu dataset. Cosmopedia-v2 is a high-quality dataset generated by a model based on 34,000 topics defined by BISAC book classifications. Additionally, open-web-math[10] and StackOverflow[11] are sourced from high-quality mathematical texts available online and posts from StackOverflow, respectively.

Hyperparameters For the $135\text{M}_{\text{SmolLM}}$ and $360\text{M}_{\text{SmolLM}}$ models, the global batch size is set to 64. The models are trained for a total of 18,000 steps. The learning rate is set to 0.0001, with a warmup phase spanning the first 900 steps, following a linear warmup strategy. The learning rate decay begins after 16,200 steps, extending over the final 1,800 steps of the training process. A 1-sqrt decay strategy is applied to adjust the learning rate, ensuring a smooth and gradual reduction as training progresses.The fine-tuning pro-

[8] https://huggingface.co/blog/smollm

[9] https://hf-mirror.com/datasets/HuggingFaceTB/smollm-corpus

[10] https://huggingface.co/datasets/open-web-math/open-web-math

[11] https://huggingface.co/datasets/bigcode/stackoverflow-clean

cess for $135\text{M}_{\text{SmolLM}}$ takes about 6 hours on 4 RTX3090 GPUs with a maximum sequence length 2048.When training the $1\text{B}7_{\text{SmolLM}}$ and $7\text{B}_{\text{Llama2}}$ models, the global batch size is set to 256, and the learning rate is set to 1e-4. The training consists of 12,000 steps, with a warmup phase spanning the first 1,000 steps. The learning rate begins to decay after 10,000 steps. A linear warmup strategy is applied during the warmup phase, while a 1-sqrt decay strategy is used for the learning rate decay.We utilized 16 NVIDIA L20Y GPUs to fine-tune the LLama-7B model over a duration of 24 hours.

C Ablation Study on Partial-RoPE Dimensions

To better determine the strategy and dimensionality for partial-RoPE, we conducted an ablation study on the number of RoPE dimensions using the $135\text{M}_{\text{SmolLM}}$ model. The experimental results are presented in Table 4. By comparing the performance of four different strategies across varying dimensionalities, we observed that the low-frequency strategy, $\mathcal{S}_{\text{low}}$, suffered significant performance degradation (-14.7%) when the dimensionality was relatively low (≤ 4). In contrast, both $\mathcal{S}_{\text{uniform}}$ and $\mathcal{S}_{\text{2-norm}}$ consistently demonstrated superior performance regardless of the dimensionality. Additionally, increasing the dimensionality from 4 to 8 provided negligible performance gains. Based on these findings, we selected a dimensionality of 4 for partial-RoPE.

D LongBench Results

In this section, we present the evaluation results of $1\text{B}7_{\text{SmolLM}}$ on the LongBench benchmark. The results are documented in the Table 5.

Model		Avg.	MMLU	ARC	PIQA	HS	OBQA	WG
135M	r=32	44.25	29.82	42.05	68.34	41.03	33.20	51.07
- $\mathcal{S}_{\text{high}}$	r=2	$42.86_{-1.39}$	29.58	40.91	66.54	38.48	32.00	49.64
	r=4	$43.40_{-0.85}$	29.90	41.15	66.92	39.34	32.60	50.51
	r=8	$43.56_{-0.69}$	29.90	40.89	67.63	40.41	32.20	50.36
- $\mathcal{S}_{\text{low}}$	r=2	$37.94_{-6.31}$	26.95	33.56	60.28	31.51	27.80	47.51
	r=4	$37.76_{-6.49}$	27.11	32.06	59.79	30.68	28.40	48.54
	r=8	$42.54_{-1.71}$	29.34	39.58	67.36	37.86	32.00	49.09
- $\mathcal{S}_{\text{uniform}}$	r=2	$43.16_{-1.09}$	29.89	41.80	66.27	38.78	32.40	49.80
	r=4	$43.76_{-0.49}$	29.87	41.29	67.36	40.22	32.80	50.99
	r=8	$43.74_{-0.51}$	29.95	40.81	67.19	40.47	32.60	51.38
- $\mathcal{S}_{\text{2-norm}}$	r=2	$43.13_{-1.12}$	29.75	40.13	67.25	39.03	32.80	49.80
	r=4	$43.77_{-0.48}$	30.14	41.69	67.57	39.53	33.00	50.67
	r=8	$43.88_{-0.37}$	29.91	41.35	67.74	40.40	33.40	50.51

Table 4: The impact of positional encoding dimensionality on model performance.

Model	Precision	KV Mem.	Avg@LB
$1B7_{\text{SmolLM}}$	BF16	100.0%	18.7
	Int4_{HQQ}	-75.00%	18.6
	$\text{Int4}_{\text{Quanto}}$		18.6
	Int2_{HQQ}	-87.50%	16.3
	$\text{Int2}_{\text{Quanto}}$		13.3
$d_{kv} = 32$	BF16	-68.75%	16
	Int4_{HQQ}	-92.19%	15.9
	$\text{Int4}_{\text{Quanto}}$		15.4
$d_{kv} = 16$	BF16	-81.25%	16.5
	Int4_{HQQ}	-95.31%	16.2
	$\text{Int4}_{\text{Quanto}}$		15.6
$d_{kv} = 8$	BF16	-87.50%	15.3
	Int4_{HQQ}	-96.87%	15
	$\text{Int4}_{\text{quanto}}$		14.2

Table 5: Evaluation results of models on LongBench