TransMLA: Migrating GQA Models to MLA with Full DeepSeek Compatibility and Speedup

Fanxu Meng[1,2]*, **Pingzhi Tang**[1]*, **Zengwei Yao**[4], **Xing Sun**[3], **Muhan Zhang**[1,2]†
[1]Institute for Artificial Intelligence, Peking University
[2]State Key Laboratory of General Artificial Intelligence, Peking University
[3]Tencent, Shanghai, China
[4]Xiaomi Corp., Beijing, China
https://github.com/fxmeng/TransMLA

Abstract

Modern large-language models often face communication bottlenecks on current hardware rather than computational limitations. *Multi-head latent attention* (*MLA*) addresses this by compressing the key-value cache using low-rank matrices, while the Absorb operation prevents the KV cache from reverting to its original size, significantly boosting both training and inference speed. Despite the success of DeepSeek V2/V3/R1, most model vendors have heavily invested in optimizing GQA-based models and therefore lack strong incentives to retrain MLA-based models from scratch. In this paper, we introduce TransMLA, a framework that seamlessly converts any GQA-based pre-trained model (e.g., LLaMA, Qwen, Mixtral) into an MLA-based model. For the first time, our method enables *direct conversion of these models into a format compatible with DeepSeek's codebase*, allowing them to fully leverage DeepSeek-specific optimizations such as vLLM and SGlang. By compressing 93% of the KV cache in LLaMA-2-7B, we achieve a **10.6x speedup** with an 8K context length while maintaining meaningful output. Moreover, the model requires only **6B tokens** for fine-tuning to recover comparable performance across multiple benchmarks. TransMLA provides a practical path for migrating GQA-based models to the MLA structure, and when combined with DeepSeek's advanced optimizations—such as FP8 quantization and Multi-Token Prediction—further inference acceleration can be achieved.

1 Introduction

In recent years, Large Language Models (LLMs) have become indispensable tools for productivity. Notable models include GPT-4o OpenAI [2024], DeepSeek R1 Guo et al. [2025], Claude 3.7 Sonnet Anthropic [2024], and Gemini-2.5 Team et al. [2024a], etc. Open-source models like LLaMA-4 AI@Meta [2024], Mistral-3 Mistral [2024], Qwen-3 Qwen [2024], DeepSeek V3/R1 Liu et al. [2024a], Gemma-3 Team et al. [2024b], and Phi-4 Abdin et al. [2024] have narrowed the performance gap to closed-source alternatives. The effectiveness of LLMs primarily stems from Next Token Prediction Radford [2018], Brown et al. [2020], where tokens are predicted sequentially, with attention computed between each token and the tokens preceding it. To optimize computations, key-value pairs are cached (KV cache). However, as model sizes grow, the overhead of caching increases, leading to memory and communication bottlenecks.

To address these challenges, Group-Query Attention (GQA)Ainslie et al. [2023] was introduced to group query heads, where all query heads in each group shares a single key and value head. When

*Equal contribution.
†Corresponding author: muhan@pku.edu.cn

Preprint. Under review.

there is one group, it becomes Multi-Query Attention (MQA) Shazeer [2019], and when the number of groups equals the number of heads, it becomes Multi-Head Attention (MHA) Vaswani et al. [2017]. Both GQA and MQA reduce KV cache requirements compared to MHA, but they sacrifice model performance at the same time. Additionally, techniques such as post-training KV cache compressionXiao et al. [2024], Liu et al. [2024b], Hooper et al. [2024], Zhang et al. [2023] result in non-standard implementations that require specialized optimizations, limiting their widespread adoption.

A pre-trained KV cache compression method, Multi-Head Latent Attention (MLA), introduced in DeepSeek V2 DeepSeek-AI [2024] and extended in DeepSeek V3 DeepSeek-AI [2024] and DeepSeek R1 Guo et al. [2025], strikes a better balance between computational efficiency and model performance. These models achieve top-tier performance while reducing the training and inference costs of LLMs to a new level. Furthermore, the DeepSeek team actively contributes to the open-source ecosystem, enabling highly optimized implementation and deployment solutions for efficient use of their models.

In this paper, we first prove that MLA consistently offers *higher expressive power* than GQA under the same KV cache overhead, which theoretically explains the advantage of MLA. However, a practical obstacle preventing model vendors from switching to MLA is the substantial prior investment on GQA-based models. This motivates us to ask, can we *seamlessly convert a GQA-based pretrained model*, such as LLaMA-2&3 AI@Meta [2024] and Qwen-2.5&3 Qwen [2024], *to MLA so that we can inherit the model weights and pretraining effort*, rather than training MLA from scratch?

A key challenge in converting a GQA-based model to MLA is the presence of Rotary Position Embedding (RoPE) [Su et al., 2024] in each query-key head, which prevents the use of the *Absorb* Chang et al. [2024] operation, which is crucial in DeepSeek for switching between computation-efficient and memory-efficient modes. Inspired by DeepSeek's Decoupled RoPE strategy, we propose a solution that concentrates the positional information in keys into a small subset of embedding dimensions. The remaining dimensions, which have minimal positional information, have their RoPE removed and are merged with the values for low-rank decompositions. The key up-projection, without RoPE, can then be absorbed into the query as in DeepSeek models.

To efficiently concentrate positional information into fewer dimensions, we introduce **RoRoPE**—a novel technique that performs principal component analysis (PCA) on the key output, applies rotation across the two ends of RoPE, and consolidates the principal components of all attention heads into the dimensions of the first attention head. We theoretically prove that the product remains invariant after rotating the query and key using a matrix $\mathbf{U}$, as long as $\mathbf{U}$ satisfies two conditions: (1) rotation occurs only within the same dimension across all attention heads, and (2) the real and imaginary components of RoPE are rotated in the same manner. Additionally, by exploiting the frequency similarity between adjacent RoPE dimensions, we propose **FreqFold**, a technique that improves the concentration efficiency in the first attention head.

Finally, we observe that the magnitude of the keys, after removing RoPE, significantly exceeds that of the values. Directly applying PCA on both the keys and values tends to preserve the principal directions of the keys only, resulting in a loss of information from the Value and significant degradation in compressed model performance. To address this issue, we propose a **Balanced Key-Value** method, which first balances the norms of the keys and values and then jointly performs PCA, resulting in a notable improvement in compression performance.

The above innovations collectively form our TransMLA method. Using TransMLA, we compressed the KV cache of LLaMA-2 by 68.75%, with only a 1.65% performance drop across 6 benchmarks for training free. In contrast, a concurrent method, MHA2MLA Ji et al. [2025], experienced a 21.85% performance decline. At a compression rate of 93%, the model still maintained meaningful responses, and after training with just 6B tokens, its performance was mostly restored. We tested both the original and TransMLA models on three different hardware setups using vLLM, achieving up to a 10.6x speedup compared to the original GQA models, demonstrating the great potential of TransMLA. Moreover, the TransMLA models are fully compatible with DeepSeek's code, enjoying DeepSeek's ecosystem to accelerate inference and seamlessly integrate with various hardware and frameworks.

2 Related Work

In large language models (LLMs), the key-value (KV) cache is essential for efficient autoregressive decoding, as it stores past key and value vectors to avoid redundant computations. However, the memory consumption of the KV cache grows linearly with the sequence length, since each new token requires storing its KV pair. This increasing memory footprint can become a bottleneck when processing very long sequences. To address this challenge, various techniques have been proposed to reduce the KV cache's memory usage while maintaining model performance.

Group-Query Attention (GQA) Ainslie et al. [2023] improves KV cache efficiency by grouping query heads so that all queries in a group share a single key and value head. When the number of groups is one, GQA degenerates to Multi-Query Attention (MQA)Shazeer [2019]. Both GQA and MQA reduce KV cache memory requirements compared to MHA but often sacrifice some model performance. Multi-Head Latent Attention (MLA), introduced in DeepSeek V2 DeepSeek-AI [2024] and extended in later versions DeepSeek-AI [2024], Guo et al. [2025], is a pretrained KV cache compression method that achieves a better trade-off between computational efficiency and model accuracy. MLA significantly reduces training and inference costs while maintaining competitive performance. Tensor Product Attention (TPA) Zhang et al. [2025] takes a different approach by dynamically factorizing activations instead of static weights, enabling an order-of-magnitude reduction in inference-time KV cache size compared to MHA models. However, new attention mechanisms like TPA generally require training models from scratch, which can be costly. TransMLA does not introduce a new attention design but acts as a bridge to transform the existing GQA-based model into a MLA-based one. This transformation only requires minimal fine-tuning to restore model performance.

Another approach is to optimize the KV cache of existing pre-trained models. For example, dynamic token pruning is employed by LazyLLM Fu et al. [2024], A2SF Jo and Shin [2024], and SnapKV Li et al. [2024]. These methods selectively prune less important tokens from the KV cache. Sharing KV representations across layers, as in YONO Sun et al. [2024], MiniCache Liu et al. [2024c], and MLKV Zuhri et al. [2024], reduces memory by reusing the same KV cache across multiple layers. This can drastically lower memory usage and speed up inference. However, these optimization methods often require specialized configurations, making deployment difficult and limiting their widespread use. In contrast, TransMLA leverages the mature ecosystem of DeepSeek, allowing direct loading of the transformed model, making it easy to accelerate deployment across various DeepSeek-supported platforms.

There are two works most related to TransMLA. One is Palu Chang et al. [2024], which reduces KV cache size by applying low-rank decomposition on both the keys and values, enabling speedup through tailored optimizations. However, Palu does not specifically handle RoPE, which prevents it from using the Absorb operation during inference. Therefore, Palu needs to project the compressed representations back to their original size. This projection incurs significant computational overhead during inference, limiting the overall acceleration. Another concurrent work, MHA2MLA Ji et al. [2025], also claims to convert MHA to MLA and decouple RoPE from the main computational path. It is important to clarify that TransMLA is not simply a GQA extension of MHA2MLA—both TransMLA and MHA2MLA support MHA and GQA architectures. However, MHA2MLA determines which RoPE dimensions to remove solely based on the norms of the query and key vectors, which tends to cause larger information loss when pruning the same proportion of positions. Also, the distribution of important dimensions in MHA2MLA is uneven, requiring sparse indexing that complicates optimization and acceleration. Their work reports compression ratios of the KV cache but does not demonstrate actual inference speedup. Furthermore, MHA2MLA directly applies joint singular value decomposition to KV, resulting in higher loss compared to our balanced key-value PCA method.

3 Preliminary

3.1 Rotary Position Embedding (RoPE)

RoPE Su et al. [2024] is a position encoding method that encodes the absolute positions with different rotations and incorporates the explicit relative position dependency in the self-attention formulation. It applies different rotations to tokens in different positions to encode the position information.

Consider $\mathbf{x}_t \in \mathbb{R}^d$ to be the embedding of the t-th token with the hidden size d. The RoPE operation upon x_t produces a representation x_t^R that encodes both semantic and positional information:

$$\mathbf{x}_t^R = \text{RoPE}(\mathbf{x}_t, t) = \begin{pmatrix} \mathbf{x}_t^{(1)} \\ \mathbf{x}_t^{(2)} \\ \mathbf{x}_t^{(3)} \\ \mathbf{x}_t^{(4)} \\ \vdots \\ \mathbf{x}_t^{(d-1)} \\ \mathbf{x}_t^{(d)} \end{pmatrix} \otimes \begin{pmatrix} \cos t\theta_1 \\ \cos t\theta_1 \\ \cos t\theta_2 \\ \cos t\theta_2 \\ \vdots \\ \cos t\theta_{d/2} \\ \cos t\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -\mathbf{x}_t^{(2)} \\ \mathbf{x}_t^{(1)} \\ -\mathbf{x}_t^{(4)} \\ \mathbf{x}_t^{(3)} \\ \vdots \\ -\mathbf{x}_t^{(d)} \\ \mathbf{x}_t^{(d-1)} \end{pmatrix} \otimes \begin{pmatrix} \sin t\theta_1 \\ \sin t\theta_1 \\ \sin t\theta_2 \\ \sin t\theta_2 \\ \vdots \\ \sin t\theta_{d/2} \\ \sin t\theta_{d/2} \end{pmatrix}, \tag{1}$$

where $\otimes$ denotes the element-wise multiplication of two vectors, $\mathbf{x}_t^{(i)} \in \mathbb{R}$ denotes the i-th element of $\mathbf{x}_t$, and $\theta_i = 10000^{-2(i-1)/d}$ is the i-th rotation angle. If we interpret every two elements in the embedding as a representation in the complex coordinate system, we can divide $\mathbf{x}_t$ into paired dimensions, where the odd-indexed dimensions $\mathbf{x}_t^{(2k-1)}$ represent the real parts and the even-indexed dimensions $\mathbf{x}_t^{(2k)}$ represent the imaginary parts.

3.2 Group Query Attention

Let the t-th token of the input sequence be $\mathbf{x}_t \in \mathbb{R}^D$, where D denotes the hidden dimension. To reduce the memory overhead of the KV cache, GQA divides the h query heads uniformly into g groups, with all query heads within a group sharing the same key and value vectors. Specifically, let $W^Q \in \mathbb{R}^{hd \times D}$, $\mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{gd \times D}$ and $W^O \in \mathbb{R}^{D \times hd}$ be the projection matrices for the query, key, value and output, where $d = D/h$ denotes the dimension per head. GQA first computes the concatenated queries $\mathbf{q}_t$, keys $\mathbf{k}_t$, and values $\mathbf{v}_t$, and then slices them into heads or groups for attention computation:

$$[\mathbf{q}_{t,1}; \mathbf{q}_{t,2}; ...; \mathbf{q}_{t,h}] = \mathbf{q}_t = W^Q \mathbf{x}_t, \tag{2}$$

$$[\mathbf{k}_{t,1}; \mathbf{k}_{t,2}; ...; \mathbf{k}_{t,g}] = \mathbf{k}_t = W^K \mathbf{x}_t, \tag{3}$$

$$[\mathbf{v}_{t,1}; \mathbf{v}_{t,2}; ...; \mathbf{v}_{t,g}] = \mathbf{v}_t = W^V \mathbf{x}_t, \tag{4}$$

where each $\mathbf{q}_{t,i} \in \mathbb{R}^d$ corresponds to the query vector of the i-th head, and $\mathbf{k}_{t,j}, \mathbf{v}_{t,j} \in \mathbb{R}^d$ correspond to the key and value vectors of the j-th group.

Using the notation in Section 3.1, after applying RoPE to $\mathbf{q}_{t,i}, \mathbf{k}_{t,i}$, we can obtain the attention output for the t-th token as follows:

$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \text{softmax}_j\left(\frac{{\mathbf{q}_{t,i}^R}^\top \mathbf{k}_{j,\lceil i/\frac{h}{g} \rceil}^R}{\sqrt{d}}\right) \mathbf{v}_{j,\lceil i/\frac{h}{g} \rceil}, \tag{5}$$

$$\mathbf{y}_t = W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; ...; \mathbf{o}_{t,h}]. \tag{6}$$

As we can see, in GQA, each key and value head corresponds to $\frac{h}{g}$ query heads. **When $g = h$, GQA becomes MHA, and when $g = 1$, GQA becomes Multi-Query Attention (MQA).**

3.3 Multi-Head Latent Attention

MLA saves KV cache by multiplying the matrix $W^{DKV} \in \mathbb{R}^{r_{kv} \times D}$ with the input sequence to obtain low-rank latent features. Then, it uses the matrices W^{UK} and $W^{UV} \in \mathbb{R}^{hd \times r_{kv}}$ to derive the key $\mathbf{k}$ and value $\mathbf{v}$ representations for each attention head. Additionally, MLA also decomposes W^Q to $W^{DQ} \in \mathbb{R}^{r_q \times D}$ and $W^{UQ} \in \mathbb{R}^{hd \times r_q}$, which reduces the activation memory during training. For positional embedding, MLA uses a decoupled RoPE strategy that uses additional multi-head queries $\mathbf{q}_{t,i}^R \in \mathbb{R}^{d^R}$ and a shared key $\mathbf{k}_t^R \in \mathbb{R}^{d^R}$, which are generated from $W^{QR} \in \mathbb{R}^{hd^R \times r_q}$ and $W^{KR} \in \mathbb{R}^{d^R \times d}$, to carry RoPE, where d^R denotes the per-head dimension of the decoupled queries and key.

$$
\begin{aligned}
\mathbf{c}_t^{KV} &= W^{DKV}\mathbf{x}_t, \\
[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; ...; \mathbf{k}_{t,h}^C] &= \mathbf{k}_t^C = W^{UK}\mathbf{c}_t^{KV}, \\
\mathbf{k}_t^R &= \text{RoPE}(W^{KR}\mathbf{x}_t, t), \\
\mathbf{k}_{t,i} &= [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R], \qquad (7)
\end{aligned}
\qquad
\begin{aligned}
\mathbf{c}_t^{Q} &= W^{DQ}\mathbf{x}_t, \\
[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; ...; \mathbf{q}_{t,h}^C] &= \mathbf{q}_t^C = W^{UQ}\mathbf{c}_t^{Q}, \\
[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; ...; \mathbf{q}_{t,h}^R] &= \mathbf{q}_t^R = \text{RoPE}(W^{QR}\mathbf{c}_t^{Q}, t), \\
\mathbf{q}_{t,i} &= [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R]. \qquad (8)
\end{aligned}
$$

MLA supports switching between two computational paradigms tailored for different stages. During the compute-intensive training phase, it operates in a paradigm similar to standard MHA, where the computational overhead is slightly lower than that of conventional MHA, as shown in Equation 9. For communication-intensive inference, it can seamlessly switch to a paradigm resembling MQA, as described in Equation 10. In this inference paradigm, the latent features function as a shared large KV head, which interacts with all query heads and output heads to produce the final output efficiently. This operation is called the Absorb operation, which is crucial for accelerating inference speed.

$$
\begin{aligned}
[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; ...; \mathbf{v}_{t,h}^C] &= \mathbf{v}_t^C = W^{UV}\mathbf{c}_t^{KV}, \\
\mathbf{o}_{t,i} &= \sum_{j=1}^{t} \text{softmax}_j(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d+d^R}})\mathbf{v}_{j,i}^C, \\
\mathbf{y}_t &= W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; ...; \mathbf{o}_{t,h}], \qquad (9)
\end{aligned}
\qquad
\begin{aligned}
\hat{\mathbf{q}}_{t,i} &= [W_i^{UK\top}\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R], \quad \hat{\mathbf{k}}_t = [\mathbf{c}_t^{KV}; \mathbf{k}_t^R], \\
\hat{\mathbf{o}}_{t,i} &= \sum_{j=1}^{t} \text{softmax}_j(\frac{\hat{\mathbf{q}}_{t,i}^T \hat{\mathbf{k}}_j}{\sqrt{d+d^R}})\mathbf{c}_j^{KV}, \\
\mathbf{y}_t &= W^O[W_1^{UV}\hat{\mathbf{o}}_{t,1}; ...; W_h^{UV}\hat{\mathbf{o}}_{t,h}], \qquad (10)
\end{aligned}
$$

where $W_i^{\{UK,UV\}}$ denotes slices of the projection matrices corresponding to the i-th attention head.

One of the main contributions of this paper is the seamless support for the Absorb operation, significantly enhancing inference speed.

4 TransMLA

In this section, we formally introduce TransMLA, beginning with two key observations:

1. Given the same amount of KV caches, MLA consistently exhibits greater expressive capacity than GQA. The full proof is provided in Appendix A. In summary, it demonstrates that any GQA model can be reformulated as an MLA model by introducing an additional projection matrix. When RoPE is present, the MLA should be expressed in the 'absorbed' form. However, the reverse transformation is not always possible, which implies that MLA has greater expressive power than GQA.

2. Acceleration can only be achieved when MLA operates with fewer KV caches. Although it is feasible to construct an MLA-equivalent representation of a GQA model, accelerating inference requires reducing KV caches. To address this, we transform a GQA-based model into a DeepSeek-style architecture, ensuring full compatibility with the DeepSeek inference ecosystem.

4.1 Merging All Key Heads as One

Before compressing the KV cache, it is necessary to merge all groups of key-value heads from the GQA into a single latent head. For each query head i, we introduce $W_i^{UK} \in \mathbb{R}^{d \times gd}$ with the group index $j = \left\lceil \frac{i}{h/g} \right\rceil - 1$, and initialize the matrix $W_i^{UK}[:, jd : (j+1)d]$ to be I_d (identity matrix of shape $d \times d$), with all other elements set to 0. (We adopt the matrix indexing notation used in Python and PyTorch, and will continue to do so throughout this work without further explanation.) This initialization allows the projection of the key's latent representation to be simultaneously mapped onto multiple query heads, with only the corresponding key head being multiplied by the appropriate query head. Similarly, during the computation of the multiple heads of the values, the attention scores for each head are multiplied accordingly. To keep the output unchanged, we similarly initialize W_i^{UV} so that only the corresponding value head is an identity mapping, while all other elements are set to zero. Since we have now merged all the key heads into a single head, in order to ensure that this transformation is equivalent to the original, we also merge the RoPE operations from different heads into one large RoPE operation, denoted as $\widehat{\text{RoPE}}$, applied to the entire merged key head. Since the original RoPE operation in GQA is the same for each head, $\widehat{\text{RoPE}}$ simply applies the same RoPE

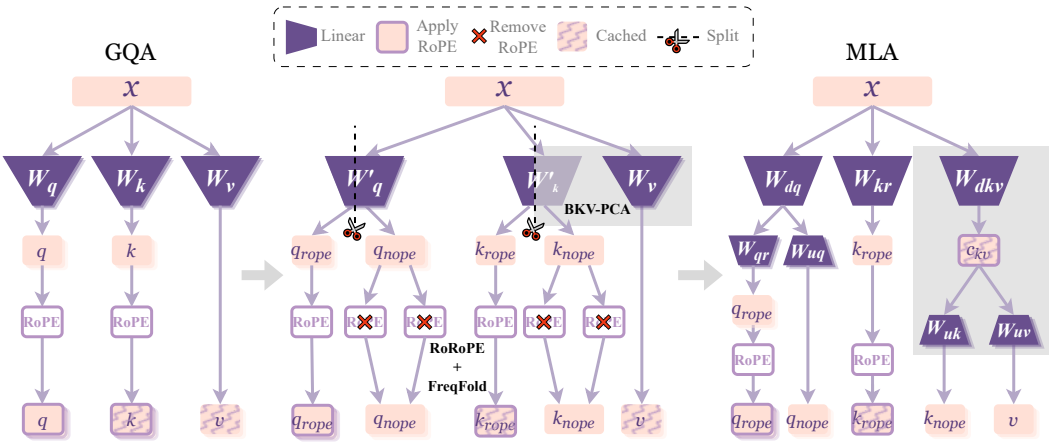


Figure 1: The pipeline for transferring GQA-based models to MLA involves applying RoRoPE to concentrate positional information in the first head, enhanced with FreqFold. KV norms are balanced before performing low-rank decomposition.

operation repeatedly for every d dimensions. In this way, the computation of GQA attention is transformed into the following form:

$$[\mathbf{c}_t^K; \mathbf{c}_t^V] = \mathbf{c}_t^{KV} = W^{DKV}\mathbf{x}_t, \quad W^{DKV} = \begin{pmatrix} W^K \\ W^V \end{pmatrix} \in \mathbb{R}^{2gd \times D} \tag{11}$$

$$[\mathbf{q}_{t,1}; \mathbf{q}_{t,2}; ...; \mathbf{q}_{t,h}] = \mathbf{q}_t = W^Q \mathbf{x}_t, \quad W^Q \in \mathbb{R}^{hd \times D} \tag{12}$$

$$\hat{\mathbf{q}}_{t,i}^R = \widehat{\text{RoPE}}\left(\left(W_i^{UK}\right)^\top \mathbf{q}_{t,i}, t\right), \quad \hat{\mathbf{k}}_t^R = \widehat{\text{RoPE}}(\mathbf{c}_t^K, t) \tag{13}$$

$$\hat{\mathbf{o}}_{t,i} = \sum_{j=1}^{t} \text{softmax}_j \left(\frac{\left(\hat{\mathbf{q}}_{t,i}^R\right)^\top \hat{\mathbf{k}}_j^R}{\sqrt{d}} \right) \mathbf{c}_j^V, \tag{14}$$

$$\mathbf{y}_t = W^O [W_1^{UV} \hat{\mathbf{o}}_{t,1}; ...; W_h^{UV} \hat{\mathbf{o}}_{t,h}]. \tag{15}$$

It is evident that the total KV cache size remains unchanged since we still need to store $\mathbf{c}_t^{KV} \in \mathbb{R}^{2gd}$ for each token, which is the same as in the original GQA model. However, the dimension of each attention head has increased by a factor of g, and the introduction of new parameters W_i^{UK} and W_i^{UV} leads to higher computational costs. To achieve actual acceleration in the transformed MLA, compressing the KV cache is therefore essential. By merging multiple KV heads, we can better identify shared principal components and represent the KV cache in a lower-dimensional latent space. Moreover, merging multiple key heads is crucial for efficiently decoupling RoPE in the subsequent steps.

4.2 Rotating Queries and Keys to Minimize Transformation Loss Towards Decoupled RoPE

In Equation 14, the term $\left(\hat{\mathbf{q}}_{t,i}^R\right)^\top \hat{\mathbf{k}}_j^R$ can be expressed as the sum of inner products over paired dimensions across multiple attention heads, incorporating positional information:

$$\left(\hat{\mathbf{q}}_{t,i}^R\right)^\top \hat{\mathbf{k}}_j^R = \sum_{l=1}^{d/2} \left[\hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \hat{\mathbf{q}}_{t,i}^{[2l::d]}\right]^{R^\top} \left[\hat{\mathbf{k}}_j^{[2l-1::d]}; \hat{\mathbf{k}}_j^{[2l::d]}\right]^R. \tag{16}$$

Here, the notation $[2l :: d]$ is inspired by Python slicing syntax and denotes selecting elements starting from the $(2l)$-th dimension, then taking every d-th element thereafter until the end of the vector. The vector $\left[\hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \hat{\mathbf{q}}_{t,i}^{[2l::d]}\right]^R$ thus has dimension $2g$.

Since multiple key heads are concatenated into a single head and each head shares the same RoPE, the real and imaginary components corresponding to the l-th 2D subspace (i.e., the paired two dimensions) of RoPE within each original attention head can be expressed as follows:

$$\left[\hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \hat{\mathbf{q}}_{t,i}^{[2l::d]}\right]^R = \cos t\theta_l \left[\hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \hat{\mathbf{q}}_{t,i}^{[2l::d]}\right] + \sin t\theta_l \left[-\hat{\mathbf{q}}_{t,i}^{[2l::d]}; \hat{\mathbf{q}}_{t,i}^{[2l-1::d]}\right], \tag{17}$$

$$\left[\hat{\mathbf{k}}_{j}^{[2l-1::d]}; \hat{\mathbf{k}}_{j}^{[2l::d]}\right]^R = \cos j\theta_l \left[\hat{\mathbf{k}}_{j}^{[2l-1::d]}; \hat{\mathbf{k}}_{j}^{[2l::d]}\right] + \sin j\theta_l \left[-\hat{\mathbf{k}}_{j}^{[2l::d]}; \hat{\mathbf{k}}_{j}^{[2l-1::d]}\right]. \tag{18}$$

When the concatenated real and imaginary components of $\hat{\mathbf{q}}_{t,i}$ and $\hat{\mathbf{k}}_j$ within the l-th 2-dimensional subspace are multiplied by an orthogonal matrix $\mathbf{U}_l \in \mathbb{R}^{g\times g}$, the inner product with RoPE applied remains invariant. Specifically,

$$\sum_{l=1}^{d/2} \left(\left[\mathbf{U}_l\hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \mathbf{U}_l\hat{\mathbf{q}}_{t,i}^{[2l::d]}\right]\right)^{R^\top} \left(\left[\mathbf{U}_l\hat{\mathbf{k}}_{j}^{[2l-1::d]}; \mathbf{U}_l\hat{\mathbf{k}}_{j}^{[2l::d]}\right]\right)^{R} = \hat{\mathbf{q}}_{t,i}^{R^\top}\hat{\mathbf{k}}_{j}^{R}. \tag{19}$$

This demonstrates that the rotational transformation $\mathbf{U}_l$ preserves the RoPE-based inner product structure. A detailed proof of this property is provided in the Appendix B.

We apply Principal Component Analysis (PCA) to the attention heads in the context of RoPE, introducing a method we call RoRoPE. Using a small dataset, we extract the key output, compute its principal component projection matrices $\{\mathbf{U}_l\}_{l\in\{1,\ldots,d/2\}}$, and rotate both W^K and W^{UK} (as shown in Eq13, $\hat{\mathbf{q}}_{t,i}$ and $\hat{\mathbf{k}}_j$ are generated from W^{UK} and W^K respectively, so rotating $\hat{\mathbf{q}}_{t,i}$ and $\hat{\mathbf{k}}_j$ can be achieved by rotating W^{UK} and W^K) using a similar procedure as described above. This rotation effectively concentrates the essential information into the first few heads. As an equivalent transformation, instead of discarding all non-principal components of the key, we remove their RoPE encoding while preserving positional information within the principal components.

To enable the transformed model to utilize standard RoPE, we represent the principal component information for corresponding positions across other heads using the dimensions of the first attention head. However, using a one-dimensional space for all positional information proves limiting. To address this, we exploit the similar frequencies of adjacent dimensions in RoPE, treating them as equivalent positions. This allows us to use multiple dimensions within a single attention head to represent positional information, a technique we refer to as **FreqFold**. Additional information on FreqFold can be found in Appendix C.

4.3 A Balanced Approach to Joint Low-Rank Approximation of NoPE Keys and Values

In the previous section, we split the key heads into one carrying positional information and the others without positional information, achieving minimal loss. We then apply Principal Component Analysis (PCA) jointly on the values and the non-positional components of the keys (i.e. NoPE-Key), using activations collected from a small calibration dataset, thereby compressing the projection matrices into a low-rank latent space. However, we observed that although the principal components of the keys were effectively separated with RoRoPE, the norm of the residual key features remained significantly larger than that of the Value. This imbalance caused the direct decomposition to favor principal component directions dominated by the keys.

To mitigate this, we scale W^{DK} by dividing it by

$$\alpha = \frac{\mathbb{E}_t[\|W_{\text{NoPE}}^{DK}\mathbf{x}_t\|_2]}{\mathbb{E}_t[\|W^{DV}\mathbf{x}_t\|_2]} \tag{20}$$

and correspondingly scale W^{UK} by multiplying it by α. Here, $W_{\text{RoPE}}^{DK} \in \mathbb{R}^{d\times D}$, $W_{\text{NoPE}}^{DK} \in \mathbb{R}^{(g-1)d\times D}$ represent the parts of W^{DK} obtained after the operations described in the previous section, where W_{RoPE}^{DK} corresponds to one head that uses RoPE, and W_{NoPE}^{DK} corresponds to the remaining heads that do not use RoPE.

This transformation is mathematically equivalent and does not affect the overall model outputs, while significantly enhancing the effectiveness of KV cache compression in subsequent steps. More details is provided in the Appendix D.

Table 1: Commonsense reasoning ability of two LLMs with TransMLA compared to MHA2MLA. The six benchmarks include MMLU (Hendrycks et al. [2021]), ARC easy and challenge (ARC, Clark et al. [2018]), PIQA (Bisk et al. [2020]), HellaSwag (HS, Zellers et al. [2019]), OpenBookQA (OBQA, Mihaylov et al. [2018]), and Winogrande (WG, Sakaguchi et al. [2021]). **Tokens** refers to the number of tokens used for further training after the TransMLA conversion. A value of 0 indicates that the model was evaluated immediately after conversion, without any fine-tuning.

Model	Tokens	KV Mem.	Avg.	MMLU	ARC	PIQA	HS	OBQA	WG
SmolLM-1.7B	1T	–	55.90	39.27	59.87	75.73	62.93	42.80	54.85
- MHA2MLA	0	-68.75%	40.97	27.73	41.96	63.00	29.19	34.40	49.53
		-87.50%	34.01	25.32	27.15	51.36	25.47	26.20	48.54
	6B	-68.75%	54.76	38.11	57.13	76.12	61.35	42.00	53.83
		-87.50%	53.61	37.17	55.50	74.86	58.55	41.20	54.38
- TransMLA	0	-68.75%	51.95	35.70	55.68	73.94	53.04	39.80	53.51
		-87.50%	44.12	29.97	41.72	66.87	41.15	34.80	50.28
	300M	-68.75%	55.24	38.60	58.95	74.97	61.52	43.00	54.38
	1B	-87.50%	54.01	37.24	56.32	74.81	60.08	42.40	53.20
LLaMA-2-7B	2T	–	59.85	41.43	59.24	78.40	73.29	41.80	64.96
- MHA2MLA	0	-68.75%	37.90	25.74	32.87	59.41	28.68	28.60	52.09
		-87.50%	32.70	25.41	25.79	50.60	26.52	19.40	48.46
	6B	-68.75%	59.51	41.36	59.51	77.37	71.72	44.20	62.90
		-87.50%	58.96	40.39	59.29	77.75	69.70	43.40	63.22
- TransMLA	0	-68.75%	58.20	39.90	57.66	77.48	70.22	41.00	62.90
		-87.50%	51.19	34.39	45.38	71.27	60.73	37.40	57.93
		-92.97%	43.26	28.93	36.32	63.38	45.87	31.60	53.43
	500M	-68.75%	59.82	40.87	59.18	77.91	71.82	45.20	63.93
	3B	-87.50%	59.36	40.77	58.84	78.18	71.28	43.60	63.46
	6B	-92.97%	58.68	40.82	59.72	76.55	69.97	43.60	61.40

5 Experiment

5.1 Main Experiment

In this section, we present our main experimental results. Following the experimental setup of MHA2MLA, we converted two models—smolLM 1.7B and Llama 2 7B—into the MLA architecture. We evaluated the models' performance on six benchmarks at three stages: before conversion, immediately after conversion without further training, and after conversion followed by training. For the training process, we used a subset of the pretraining corpus used for the smolLM model. The fine-tuned results of MHA2MLA are taken directly from the original paper. Our experiments were conducted on an 8-GPU machine, each GPU having 40GB of memory and delivering 312 TFLOPS of FP16 compute power. Detailed experimental hyperparameter settings are provided in the Appendix E.

From Table 1, we observe that TransMLA efficiently facilitates architecture migration across various models and KV cache compression ratios. Notably, the untrained performance of MLA models initialized with TransMLA shows minimal degradation in capability compared to the original models—significantly less than the degradation observed with MHA2MLA under equivalent KV cache compression. In fact, using TransMLA to compress the KV cache of Llama 2 7B to just 7.03% of its original size still results in better performance than MHA2MLA's compression to 31.25% on the same model. This highlights the effectiveness of our proposed techniques includes RoRoPE, FreqFold and activation-based balanced KV low-rank factorization.

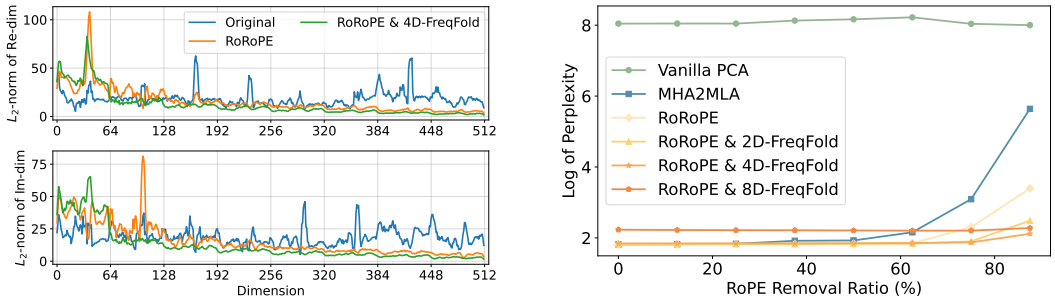
The low-loss transformation achieved by TransMLA enables us to recover the original model performance with minimal training overhead. As shown in the table, TransMLA achieves stronger performance than MHA2MLA-6B while using significantly fewer training tokens. For instance, when transforming smolLM 1.7B and compressing the KV cache to 31.25% of its original size, we only need 4.9% of the training data used by MHA2MLA and 2 hours training to surpass its performance.

5.2 Key Norm Analysis Reveals the Impact of RoRoPE and FreqFold

In this section, we conduct a detailed analysis of the distribution of key activations in the attention module to demonstrate the effectiveness of our proposed methods.

Figure 2a presents the average L2 norm of each key dimension in the first attention layer of the LLaMA 3 8B model, computed on a subset of the WikiText-2 Merity et al. [2016] dataset. We compare the original model (in blue), the model transformed using our RoRoPE equivalence method (in orange), and the further approximated model using 4D FreqFold (in green). The top and bottom halves of the plot correspond to pairs of dimensions that share the same rotation angle in RoPE, which we refer to as the real (Re-dim) and imaginary (Im-dim) dimensions.

We observe that the original model exhibits a highly *uneven norm distribution* across key dimensions, with numerous outliers. This suggests that naively removing RoPE from certain heads would likely result in significant performance degradation. After applying our RoRoPE transformation, as shown by the orange line, the key dimensions with large norms are nearly all concentrated to the first two heads (dimension 0-128). Further applying the 4D FreqFold approximation compresses the tail (i.e., higher-index dimensions) even more, leading to an even sharper concentration of high-norm key dimensions. This concentrated structure is highly beneficial for the subsequent RoPE removal step as shown in Figure 2b.



(a) Key norms of the first layer.

(b) RoPE removal results, evaluated on WikiText-2.

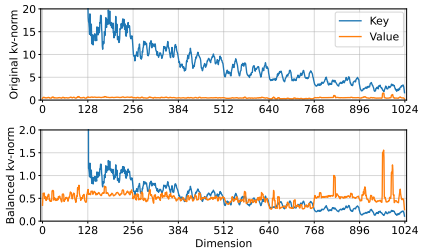
Figure 2: Visualization of key norms and RoPE removal results on LLaMA 3 8B model. The top and bottom halves of the left figure correspond to pairs of dimensions that share the same rotation angle in RoPE, which we refer to as the real (Re-dim) and imaginary (Im-dim) dimensions.

In Figure 2b, we present the log-perplexity of the LLaMA 3 8B model on WikiText-2 as RoPE components are progressively removed. We observe that our proposed RoRoPE method *significantly outperforms* MHA2MLA's per-head dimension selection strategy, especially at high removal ratio. Furthermore, incorporating similar-dimension approximation leads to even better performance under extreme removal rates. At 90% removal ration, RoRoPE + 4D-FreqFold still maintains a log-perplexity about 2, while MHA2MLA reaches nearly 6, which no longer generates meaningful outputs. At the same time, we observe that overly aggressive FreqFold (i.e., using too many dimensions) can degrade performance, as the loss introduced by approximation of nearby dimensions can outweigh the benefit in concentrating the principal components. This figure suggests that for LLaMA 3 8B, the sweet spot lies in applying RoRoPE combined with 4D FreqFold.

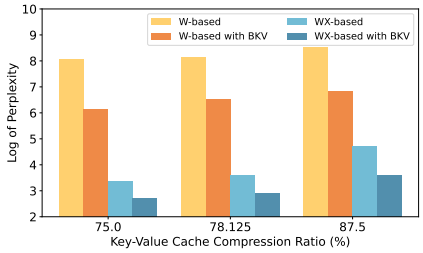
5.3 Key-Value Norm Disparity Motivates KV Balancing

In Figure 3a, we visualize the norm magnitudes of the key and value activations in the first layer of LLaMA 3 8B before and after KV balancing. Note that both the key and value shown here are activations after applying the RoRoPE principal component concentration, and the first head of the key—reserved for RoPE—is excluded. As a result, the value and the remaining key components shown in the figure are precisely the elements we aim to compress jointly into a lower-dimensional space via PCA.

It is evident that even after removing the first head with the highest norm, the overall norm of the key remains significantly larger than that of the value. This norm disparity poses a substantial challenge for joint compression into a shared latent space. In particular, such imbalance can bias the PCA



(a) Norm magnitude of keys and values before and after KV balancing is applied.



(b) Comparison of different methods of KV cache compression, with and without KV balancing.

Figure 3: Visualization of the norms of keys and values for the first layer of LLaMA 3 8B and the perplexity results after joint low-rank compression of keys and values under WikiText-2. Here, **W-based** and **WX-based** refer to PCA applied on the attention weights and the activation outputs, respectively. **BKV** denotes the application of KV balancing.

toward directions aligned with the key, rather than the value, leading to suboptimal representation of the value components.

The lower part of Figure 3a shows the norm distribution of keys and values after applying KV balancing. At this point, the norms of keys and values become more aligned, which is beneficial for performing joint PCA. This observation is further supported by the results in Figure 3b, where KV balancing consistently reduces the loss incurred by jointly applying low-rank approximation to keys and values—whether the PCA is based on weights or on activations. Figure 3b also demonstrates that activation-based PCA yields significantly better results than weight-based PCA.

5.4 Hardware-Agnostic Inference Speedup with TransMLA

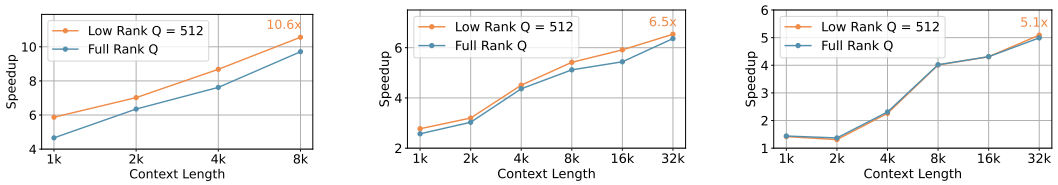
By converting MHA/GQA models into MLA models that are fully compatible with the DeepSeek codebase and compressing the KV cache, TransMLA enables us to leverage all optimizations and tooling available in DeepSeek. Using the vLLM framework, we achieve substantial real-world inference speedups.

In Figure 4, we benchmarked the inference performance of an MLA model—with a 92.97% reduction in KV cache size—on three consumer-grade AI accelerators with different compute capabilities and memory sizes: 165.2 TFLOPS with 24GB memory, 312 TFLOPS with 40GB memory, and 320 TFLOPS with 64GB memory. The figure shows the inference speedup of the MLA model relative to the original MHA model. Low-rank Q and Full-rank Q indicate whether the query projections were also compressed. Context length represents the total sequence length (i.e., context length plus generated tokens).

Our experiments show that the inference speedup of MLA models increases as the context length grows, which aligns with our expectations. Since the primary performance gain of MLA stems from KV cache compression, longer contexts lead to more substantial savings and thus higher speedups. Remarkably, for the 8K context window on the first hardware platform, the TransMLA-transformed model achieves an impressive **10.6x inference acceleration**. To the best of our knowledge, the MHA2MLA method has not reported any inference speedup results.

6 Conclusion, Limitation and Future Work

In this work, we demonstrate that the expressive power of TransMLA is stronger than GQA under the same KV cache. To help existing GQA transition to the MLA structure with minimal cost, we propose the TransMLA method. By using the RoRoPE method, the multi-head KV positional information is concentrated into the first head, and FreqFold further enhances this extraction effect. The positional information of the remaining query-key heads is removed, and the Balance KV norm method is used to jointly compress the values and the remaining heads of keys. TransMLA can convert models such as LLaMA and Qwen into MLA-based models, and the converted model incurs very little loss



(a) 165.2 TFLOPS | 24GB (b) 312 TFLOPS | 40GB (c) 320 TFLOPS | 64GB

Figure 4: Inference speedups with TransMLA comparing to the original LLaMA2 7B model on three consumer-grade AI accelerators. **Low-rank Q** and **Full-rank Q** indicate whether the query projections were also compressed. **Context length** represents the total sequence length.

compared to the original model, with performance being recoverable through training with only a few tokens. Additionally, TransMLA can easily leverage the DeepSeek ecosystem for accelerated inference, achieving significant throughput improvements across various hardware platforms.

Although TransMLA significantly reduces the loss introduced by decoupling RoPE via RoRoPE and FreqFold, and alleviates KV cache compression loss through KV balancing, the KV balancing technique itself is relatively trivial. Are there alternative, more powerful mathematical tools that can better handle the disparity between the norm of the keys and values and deliver improved performance at the same compression rate, thereby enabling truly training-free conversion? Furthermore, TransMLA needs to be validated across a broader range of models, and should be integrated with pruning, quantization, token selection, and other optimization techniques to fully explore the upper bounds of inference acceleration.

References

OpenAI. Hello GPT-4o, 2024. URL `https://openai.com/index/hello-gpt-4o/`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Anthropic. Claude 3.5 sonnet, 2024. URL `https://www.anthropic.com/news/claude-3-5-sonnet`.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024a.

AI@Meta. Llama 3 model card, 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Mistral. Cheaper, better, faster, stronger: Continuing to push the frontier of ai and making it accessible to all, 2024. URL `https://mistral.ai/news/mixtral-8x22b`.

Qwen. Qwen2.5: A party of foundation models, 2024. URL `https://qwenlm.github.io/blog/qwen2.5`.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

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. *arXiv preprint arXiv:2410.10819*, 2024.

Zhiyang Liu, Dong Zhang, Xinyi Li, and Ji Wu. Kivi: Quantized key-value representation for efficient long-context transformers. *arXiv preprint arXiv:2402.06732*, 2024b.

James Hooper, Li Dai, Zhen Zhang, and Seung-Hwan Lee. Kvquant: Quantization for efficient key-value caching in transformer models. *arXiv preprint arXiv:2402.12345*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434, 2024. URL `https://doi.org/10.48550/arXiv.2405.04434`.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 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: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling deepseek's multi-head latent attention in any transformer-based llms. *arXiv preprint arXiv:2502.14837*, 2025.

Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Zhen Qin, Yang Yuan, Quanquan Gu, and Andrew Chi-Chih Yao. Tensor product attention is all you need. *arXiv preprint arXiv:2501.06425*, 2025.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024.

Hyun-rae Jo and Dongkun Shin. A2sf: Accumulative attention scoring with forgetting factor for token pruning in transformer decoder. *arXiv preprint arXiv:2407.20485*, 2024.

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. *arXiv preprint arXiv:2404.14469*, 2024.

Tian Sun, Li Zhang, and Shuang Wu. You only need one: Efficient kv sharing across transformer layers. *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2024.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *arXiv preprint arXiv:2405.14366*, 2024c.

Zayd Muhammad Kawakibi Zuhri, Muhammad Farid Adilazuarda, Ayu Purwarianti, and Alham Fikri Aji. Mlkv: Multi-layer key-value heads for memory efficient transformer decoding. *arXiv preprint arXiv:2406.09297*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 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, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *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, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *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, 2018. doi: 10.18653/V1/D18-1260. URL https://doi.org/10.18653/v1/d18-1260.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Smollm-corpus, 2024. URL https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024a. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder 2 and the stack v2: The next generation, 2024b.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Stack Overflow. Stack overflow, 2025. URL https://stackoverflow.com. Accessed: 2025-05-21.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025a.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025b. URL https://arxiv.org/abs/2502.02737.

A Enhanced Expressive Power of MLA with Decoupled RoPE

A.1 Introduction

This section provides a theoretical analysis to demonstrate that Multi-Head Latent Attention (MLA) with decoupled Rotary Position Embedding (RoPE), as described in Section 3.3 of the main paper, possesses greater expressive power than Grouped-Query Attention (GQA) (Section 3.2). This analysis assumes **comparable KV cache sizes and number of query heads**.

Our primary argument focuses on the core projection mechanisms that generate queries, keys, and values, abstracting away from the specifics of RoPE application initially. We first present the following proposition concerning the relative expressiveness of these core mechanisms:

Proposition 1. *Given the same KV cache size and number of query heads, the expressiveness of the core attention projection mechanisms follows the order:* $\text{GQA} < \text{MLA}_{\text{Factorized}} < \text{MQA}$.

Here, $\text{MLA}_{\text{Factorized}}$ refers to an attention mechanism employing low-rank factorization for its key and value projections, representing the content-processing aspect of the full MLA. It is important to note that in the proposition, the query projection in $\text{MLA}_{\text{Factorized}}$ does not undergo low-rank factorization; this differs from the full MLA, where the query is also factorized. After proving this proposition, we will discuss how the full MLA architecture, which incorporates such an $\text{MLA}_{\text{Factorized}}$ core for its content components and an MQA core for its decoupled RoPE components, is thereby more expressive than GQA. For this analysis, we primarily consider the impact of the architectural structure on representational capacity, **setting aside the direct effects of RoPE itself** on the expressiveness comparison between the fundamental GQA, MLA-Factorized, and MQA structures.

A.2 Proof of Proposition 1

Let D be the hidden dimension of the input token $\mathbf{x}_t \in \mathbb{R}^D$, h be the number of query heads, and $d = D/h$ be the dimension per head. In GQA, query heads are divided into g groups. For fair KV cache comparison, the latent dimension for keys and values in $\text{MLA}_{\text{Factorized}}$ (r_{kv}) and the head dimension of MQA will be related to gd. Specifically, if the KV cache per token in GQA is $2gd$ for both keys and values, then in $\text{MLA}_{\text{Factorized}}$, $r_{kv} = 2gd$, and in MQA, the head dimension is also $2gd$; this ensures the KV cache sizes are aligned.

A.2.1 GQA $\leq$ MLA$_{\text{Factorized}}$

In GQA, query head $\mathbf{q}_{t,i}$ attends to key $\mathbf{k}_{j,\lceil i/(h/g)\rceil}$ and value $\mathbf{v}_{j,\lceil i/(h/g)\rceil}$. The GQA key projection $W^K \in \mathbb{R}^{gd\times D}$ produces g distinct key vectors $[\mathbf{k}_{t,1};\ldots;\mathbf{k}_{t,g}]$. Similarly, $W^V \in \mathbb{R}^{gd\times D}$ produces value vectors. We define effective per-query-head projection matrices $W'^K \in \mathbb{R}^{hd\times D}$ and $W'^V \in \mathbb{R}^{hd\times D}$ for GQA:

$$W'^K = \begin{pmatrix} W_1'^K \\ \vdots \\ W_h'^K \end{pmatrix}, \text{where } W_i'^K = W^K_{\lceil i/(h/g)\rceil}, \tag{21}$$

$$W'^V = \begin{pmatrix} W_1'^V \\ \vdots \\ W_h'^V \end{pmatrix}, \text{where } W_i'^V = W^V_{\lceil i/(h/g)\rceil}. \tag{22}$$

Here, W_k^K is the k-th $d \times D$ block of W^K. Thus, $\mathbf{k}'_{j,i} = W_i'^K \mathbf{x}_j = \mathbf{k}_{j,\lceil i/(h/g)\rceil}$, and similarly for values. The matrices W'^K and W'^V have ranks at most gd.

An $\text{MLA}_{\text{Factorized}}$ mechanism generates keys via $\mathbf{k}_{j,i} = (W^{UK}(W^{DKV}\mathbf{x}_j))_i$, where $W^{DKV} \in \mathbb{R}^{r_{kv}\times D}$ and $W^{UK} \in \mathbb{R}^{hd\times r_{kv}}$. A similar formulation applies for values with $W^{UV} \in \mathbb{R}^{hd\times r_{kv}}$.

To demonstrate expressive capability, GQA $\leq \text{MLA}_{\text{Factorized}}$, we set $r_{kv} = 2gd$. Let $W^{DKV} = \begin{pmatrix} W^K \\ W^V \end{pmatrix} \in \mathbb{R}^{2gd\times D}$. We seek $W^{UK}, W^{UV} \in \mathbb{R}^{hd\times 2gd}$ such that $W'^K = W^{UK}W^{DKV}, W'^V =$

$W^{UV}W^{DKV}$. This is achieved by setting $W_i^{UK}, W_i^{UV} \in \mathbb{R}^{d\times 2gd}$ (the block for head i) as selector matrices:

$$W_i^{UK} = [\underbrace{\mathbf{0}_{d\times d}, \ldots, \mathbf{0}_{d\times d}}_{k-1 \text{ blocks}}, \mathbf{I}_{d\times d}, \underbrace{\mathbf{0}_{d\times d}, \ldots, \mathbf{0}_{d\times d}}_{2g-k \text{ blocks}}], \tag{23}$$

$$W_i^{UV} = [\underbrace{\mathbf{0}_{d\times d}, \ldots, \mathbf{0}_{d\times d}}_{g+k-1 \text{ blocks}}, \mathbf{I}_{d\times d}, \underbrace{\mathbf{0}_{d\times d}, \ldots, \mathbf{0}_{d\times d}}_{g-k \text{ blocks}}], \tag{24}$$

where $k = \lceil i/(h/g) \rceil$. Thus, GQA's key/value generation can be replicated by an $\mathrm{MLA}_{\mathrm{Factorized}}$ model with $r_{kv} = 2gd$ and specific sparse structures for W^{UK} and W^{UV}. The KV cache size $2gd \times$ (sequence length) is preserved since we will be caching $\mathbf{c}_j^{KV} = W^{DKV}\mathbf{x}_j \in \mathbb{R}^{2gd}$. On that account, the theoretical expressive power of GQA is less than or equal to that of $\mathrm{MLA}_{\mathrm{Factorized}}$ given the same KV cache size.

A.2.2 $\mathbf{MLA_{Factorized}} \leq \mathbf{MQA}$

Consider an MLA-Factorized model where queries are $\mathbf{q}_{t,i} = W_i^Q\mathbf{x}_t$ (assuming $W_i^Q \in \mathbb{R}^{d\times D}$ is the i-th block of W^Q) and keys are $\mathbf{k}_{j,i} = (W_i^{UK}(W^{DKV}\mathbf{x}_j))$. The attention score for head i involves $\mathbf{q}_{t,i}^\top\mathbf{k}_{j,i}$:

$$\mathbf{q}_{t,i}^\top\mathbf{k}_{j,i} = (W_i^Q\mathbf{x}_t)^\top(W_i^{UK}(W^{DKV}\mathbf{x}_j)). \tag{25}$$

This can be rewritten as:

$$\mathbf{q}_{t,i}^\top\mathbf{k}_{j,i} = (\underbrace{(W_i^{UK})^\top W_i^Q}_{W_i'^Q}\mathbf{x}_t)^\top(W^{DKV}\mathbf{x}_j). \tag{26}$$

Let $\hat{\mathbf{q}}_{t,i} = W_i'^Q\mathbf{x}_t \in \mathbb{R}^{2gd}$ and $\mathbf{c}_j^{KV} = W^{DKV}\mathbf{x}_j \in \mathbb{R}^{2gd}$. The computation of attention output becomes:

$$\mathbf{o}_{t,i} = \sum_j \mathrm{softmax}_j(\frac{\hat{\mathbf{q}}_{t,i}^\top\mathbf{c}_j^{KV}}{\sqrt{d}})W_i^{UV}\mathbf{c}_j^{KV}, \tag{27}$$

$$\begin{aligned}\mathbf{y}_t &= W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; ...; \mathbf{o}_{t,h}] \\ &= \underbrace{W^O \begin{pmatrix} W_1^{UV} & & & \\ & W_2^{UV} & & \\ & & \ddots & \\ & & & W_h^{UV} \end{pmatrix}}_{W'^O} \begin{pmatrix} \mathrm{softmax}_j(\frac{\hat{\mathbf{q}}_{t,1}^\top\mathbf{c}_j^{KV}}{\sqrt{d}})\mathbf{c}_j^{KV} \\ \vdots \\ \mathrm{softmax}_j(\frac{\hat{\mathbf{q}}_{t,1}^\top\mathbf{c}_j^{KV}}{\sqrt{d}})\mathbf{c}_j^{KV} \end{pmatrix}.\end{aligned} \tag{28}$$

This is an MQA formulation where each modified query $\hat{\mathbf{q}}_{t,i}$ (now of dimension $2gd$) attends to a shared key and value $\mathbf{c}_j^{KV}$. This indicates that the computations within MLA-Factorized can be structured to use shared intermediate key and value representations akin to MQA's core. Thus, any MLA-Factorized model can be represented as an MQA model with a shared key/value of dimension $2gd$.

A.2.3 Strict Inequalities: $\mathbf{GQA} < \mathbf{MLA_{Factorized}} < \mathbf{MQA}$

The relationships are strict:

$\mathbf{GQA} < \mathbf{MLA_{Factorized}}$ When GQA is represented as an $\mathrm{MLA}_{\mathrm{Factorized}}$ model, the up-projection matrices W^{UK} and W^{UV} must adopt specific sparse, block-selector structures. A general $\mathrm{MLA}_{\mathrm{Factorized}}$ model imposes no such constraints; W^{UK} and W^{UV} are typically dense and fully learnable. This allows a general $\mathrm{MLA}_{\mathrm{Factorized}}$ to create h distinct key (and value) vectors by combining features from the r_{kv}-dimensional latent space in complex ways. GQA is restricted to g unique key (and value) vectors that are merely replicated h/g times. If $h > g$, $\mathrm{MLA}_{\mathrm{Factorized}}$ can generate a richer set of interaction patterns. Thus, $\mathrm{MLA}_{\mathrm{Factorized}}$ has strictly greater expressive power.

$\text{MLA}_{\text{Factorized}} < \text{MQA}$ Consider the bilinear form $\mathbf{x}_t^\top \mathbf{M} \mathbf{x}_j$ in the attention score. In $\text{MLA}_{\text{Factorized}}$, for head i, $\mathbf{M}_{MLA,i} = (W_i^Q)^\top W_i^{UK} W^{DKV}$. The maximum rank of the transformation is determined by the smallest one among the ranks of $W_i^Q \in \mathbb{R}^{d \times D}$, $W_i^{UK} \in \mathbb{R}^{d \times 2gd}$, and $W^{DKV} \in \mathbb{R}^{2gd \times D}$, which is at most d.

However, in the MQA form derived from $\text{MLA}_{\text{Factorized}}$, the rank of the interaction matrix here, $(W_i'^Q)^\top W^{DKV}$, is determined by the smallest one among the ranks of $W_i'^Q \in \mathbb{R}^{2gd \times D}$ and $W^{DKV} \in \mathbb{R}^{2gd \times D}$, which is at most $2gd$.

Since $2gd \geq d$, MQA allows for a potentially higher-rank interaction between the (modified) query and the shared key representations compared to the per-head effective rank in $\text{MLA}_{\text{Factorized}}$'s original formulation. This indicates that MQA has a greater representational capacity for the scoring mechanism.

A.3 Expressiveness of MLA with Decoupled RoPE

The full MLA architecture, as defined in Section 3.3 (main paper), employs a decoupled RoPE strategy. The query $\mathbf{q}_{t,i}$ and key $\mathbf{k}_{t,i}$ for head i (in the MHA-like training paradigm, Equation 9) are:

$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R] \tag{29}$$

$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R] \tag{30}$$

where $\mathbf{k}_t^R$ is a shared RoPE key component across all heads for token t. The bilinear attention score (numerator of the softmax argument) for head i between query at t and key at j is:

$$(\mathbf{q}_{t,i}^C)^\top \mathbf{k}_{j,i}^C + (\mathbf{q}_{t,i}^R)^\top \mathbf{k}_j^R \tag{31}$$

Let's analyze the two components of this score:

1. **Content Component Interaction**: $(\mathbf{q}_{t,i}^C)^\top \mathbf{k}_{j,i}^C$. The content keys $\mathbf{k}_{j,i}^C$ are derived from $W^{UK}(W^{DKV}\mathbf{x}_j)$. This key generation mechanism for $\mathbf{k}_{j,i}^C$ is precisely that of the $\text{MLA}_{\text{Factorized}}$ model discussed in Section A.1. As established, MLA-Factorized is strictly more expressive than GQA for the non-positional part of the representation.
2. **Positional Component Interaction**: $(\mathbf{q}_{t,i}^R)^\top \mathbf{k}_j^R$. This interaction, where h distinct query-side RoPE components $\mathbf{q}_{t,i}^R$ attend to a single, shared key-side RoPE component $\mathbf{k}_j^R$, is an MQA structure specifically for the positional information. As shown in Section A.2.3, MQA is strictly more expressive than $\text{MLA}_{\text{Factorized}}$, and by extension, GQA.

In summary, we have demonstrated that the expressive power of MLA with decoupled RoPE is stronger than that of the traditional GQA. However, it is worth noting that in the previously proven proposition, the $\text{MLA}_{\text{Factorized}}$ does not have a low-rank decomposition on the query; this differs from DeepSeek MLA. In the full MLA architecture, the query is also decomposed.

B Proof of RoPE Inner Product Invariance under Orthogonal Transformation

In this subsection, we provide a rigorous proof of Equation 19, namely:

$$\sum_{l=1}^{d/2} \left(\left[\mathbf{U}_l \hat{\mathbf{q}}_{t,i}^{[2l-1::d]}; \mathbf{U}_l \hat{\mathbf{q}}_{t,i}^{[2l::d]} \right] \right)^{R^\top} \left(\left[\mathbf{U}_l \hat{\mathbf{k}}_j^{[2l-1::d]}; \mathbf{U}_l \hat{\mathbf{k}}_j^{[2l::d]} \right] \right)^R = \hat{\mathbf{q}}_{t,i}^{R^\top} \hat{\mathbf{k}}_j^R.$$

Here, d is the dimension of each original attention head. The notation $\mathbf{q}_{t,i}^{[2l-1::d]}$ (and similarly for other terms) refers to an h-dimensional vector collecting the $(2l-1)$-th components from each of the h original attention heads. The matrix $\mathbf{U}_l$ is an $h \times h$ orthogonal matrix. The superscript R denotes the application of RoPE.

Proof. For the sake of convenience, we omit all i, j, k and let $\mathbf{q}_{x,l} = \mathbf{q}_{t,i}^{[2l-1::]}$ and $\mathbf{q}_{y,l} = \mathbf{q}_{t,i}^{[2l::]}$. These are h-dimensional vectors. Similarly, let $\mathbf{k}_{x,l} = \mathbf{k}_j^{[2l-1::]}$ and $\mathbf{k}_{y,l} = \mathbf{k}_j^{[2l::]}$.

The RoPE transformation, as defined by Equations (17) and (18) in the main text, applies as follows for a query vector at position t and key vector at position j within the l-th subspace:

For the query vector components:

$$(\mathbf{q}_{x,l})^R = \mathbf{q}_{x,l}\cos(t\theta_l) - \mathbf{q}_{y,l}\sin(t\theta_l)$$
$$(\mathbf{q}_{y,l})^R = \mathbf{q}_{x,l}\sin(t\theta_l) + \mathbf{q}_{y,l}\cos(t\theta_l)$$

For the key vector components:

$$(\mathbf{k}_{x,l})^R = \mathbf{k}_{x,l}\cos(j\theta_l) - \mathbf{k}_{y,l}\sin(j\theta_l)$$
$$(\mathbf{k}_{y,l})^R = \mathbf{k}_{x,l}\sin(j\theta_l) + \mathbf{k}_{y,l}\cos(j\theta_l)$$

We use the shorthand $c_t = \cos(t\theta_l)$, $s_t = \sin(t\theta_l)$, $c_j = \cos(j\theta_l)$, and $s_j = \sin(j\theta_l)$.

The right-hand side (RHS) of Equation (19) is given by the definition of the RoPE inner product:

$$\begin{aligned}
{\mathbf{q}_{t,i}^R}^\top \mathbf{k}_j^R &= \sum_{l=1}^{d/2} \left[(\mathbf{q}_{x,l})^R; (\mathbf{q}_{y,l})^R\right]^\top \left[(\mathbf{k}_{x,l})^R; (\mathbf{k}_{y,l})^R\right] \\
&= \sum_{l=1}^{d/2} \left(((\mathbf{q}_{x,l})^R)^\top (\mathbf{k}_{x,l})^R + ((\mathbf{q}_{y,l})^R)^\top (\mathbf{k}_{y,l})^R\right)
\end{aligned}$$

Let S_l be the l-th term in this sum:

$$\begin{aligned}
S_l &= (c_t\mathbf{q}_{x,l} - s_t\mathbf{q}_{y,l})^\top (c_j\mathbf{k}_{x,l} - s_j\mathbf{k}_{y,l}) + (s_t\mathbf{q}_{x,l} + c_t\mathbf{q}_{y,l})^\top (s_j\mathbf{k}_{x,l} + c_j\mathbf{k}_{y,l}) \\
&= c_tc_j\mathbf{q}_{x,l}^\top\mathbf{k}_{x,l} - c_ts_j\mathbf{q}_{x,l}^\top\mathbf{k}_{y,l} - s_tc_j\mathbf{q}_{y,l}^\top\mathbf{k}_{x,l} + s_ts_j\mathbf{q}_{y,l}^\top\mathbf{k}_{y,l} \\
&\quad + s_ts_j\mathbf{q}_{x,l}^\top\mathbf{k}_{x,l} + s_tc_j\mathbf{q}_{x,l}^\top\mathbf{k}_{y,l} + c_ts_j\mathbf{q}_{y,l}^\top\mathbf{k}_{x,l} + c_tc_j\mathbf{q}_{y,l}^\top\mathbf{k}_{y,l} \\
&= (c_tc_j + s_ts_j)(\mathbf{q}_{x,l}^\top\mathbf{k}_{x,l} + \mathbf{q}_{y,l}^\top\mathbf{k}_{y,l}) + (s_tc_j - c_ts_j)(\mathbf{q}_{x,l}^\top\mathbf{k}_{y,l} - \mathbf{q}_{y,l}^\top\mathbf{k}_{x,l}) \\
&= \cos((t-j)\theta_l)(\mathbf{q}_{x,l}^\top\mathbf{k}_{x,l} + \mathbf{q}_{y,l}^\top\mathbf{k}_{y,l}) + \sin((t-j)\theta_l)(\mathbf{q}_{x,l}^\top\mathbf{k}_{y,l} - \mathbf{q}_{y,l}^\top\mathbf{k}_{x,l}).
\end{aligned}$$

Now, let's analyze the left-hand side (LHS) of Equation (19). Let $\mathbf{q}'_{x,l} = \mathbf{U}_l\mathbf{q}_{x,l}$ and $\mathbf{q}'_{y,l} = \mathbf{U}_l\mathbf{q}_{y,l}$. Similarly, let $\mathbf{k}'_{x,l} = \mathbf{U}_l\mathbf{k}_{x,l}$ and $\mathbf{k}'_{y,l} = \mathbf{U}_l\mathbf{k}_{y,l}$. The l-th term of the LHS sum, denoted S'_l, is:

$$S'_l = \left(((\mathbf{q}'_{x,l})^R)^\top (\mathbf{k}'_{x,l})^R + ((\mathbf{q}'_{y,l})^R)^\top (\mathbf{k}'_{y,l})^R\right).$$

This has the same structure as S_l, just with primed variables:

$$S'_l = \cos((t-j)\theta_l)((\mathbf{q}'_{x,l})^\top\mathbf{k}'_{x,l} + (\mathbf{q}'_{y,l})^\top\mathbf{k}'_{y,l}) + \sin((t-j)\theta_l)((\mathbf{q}'_{x,l})^\top\mathbf{k}'_{y,l} - (\mathbf{q}'_{y,l})^\top\mathbf{k}'_{x,l}).$$

We need to show that the dot product terms involving primed variables are equal to their unprimed counterparts. Consider the first coefficient term:

$$\begin{aligned}
(\mathbf{q}'_{x,l})^\top\mathbf{k}'_{x,l} + (\mathbf{q}'_{y,l})^\top\mathbf{k}'_{y,l} &= (\mathbf{U}_l\mathbf{q}_{x,l})^\top(\mathbf{U}_l\mathbf{k}_{x,l}) + (\mathbf{U}_l\mathbf{q}_{y,l})^\top(\mathbf{U}_l\mathbf{k}_{y,l}) \\
&= \mathbf{q}_{x,l}^\top\mathbf{U}_l^\top\mathbf{U}_l\mathbf{k}_{x,l} + \mathbf{q}_{y,l}^\top\mathbf{U}_l^\top\mathbf{U}_l\mathbf{k}_{y,l} \\
&= \mathbf{q}_{x,l}^\top\mathbf{k}_{x,l} + \mathbf{q}_{y,l}^\top\mathbf{k}_{y,l}.
\end{aligned}$$

The last equation holds because U_l is an orthogonal matrix. This matches the corresponding term in S_l.

The same applies to the second coefficient term. In this way, we have proven that $S'_l = S_l$ for each $l \in \{1, \ldots, d/2\}$. This implies that the LHS of Equation (19) is equal to its RHS:

$$\sum_{l=1}^{d/2} \left(\left[\mathbf{U}_l\mathbf{q}_{t,i}^{[2l-1::]}; \mathbf{U}_l\mathbf{q}_{t,i}^{[2l::]}\right]\right)^{R^\top} \left(\left[\mathbf{U}_l\mathbf{k}_j^{[2l-1::]}; \mathbf{U}_l\mathbf{k}_j^{[2l::]}\right]\right)^R = {\mathbf{q}_{t,i}^R}^\top \mathbf{k}_j^R.$$

This completes the proof, demonstrating that the orthogonal transformation $\mathbf{U}_l$ applied to the h-dimensional vectors representing the l-th 2D subspace components across heads preserves the RoPE-based inner product structure. □

In practice, we take advantage of this property to find such orthogonal matrices U_l that concentrate the principal components of keys into the first head. In this way, when we remove the principal components of most heads in queries and keys, we can retain most of the positional information in the first head, significantly reducing the loss caused by removing RoPE.

Specifically, we proceed as follows: First, we run the model on a calibration dataset (Wikitext-2) to obtain the keys at each layer. Then, we perform PCA on the key activations across the dimensions corresponding to different heads. The resulting matrix of eigenvectors (ordered by the magnitude of their corresponding eigenvalues, from largest to smallest) is used as the orthogonal matrix U_l in this context. As a result, after this rotation, the principal components of the keys in each dimension are concentrated in the first few heads.

C FreqFold: Detailed Mechanism, Example, and PCA Efficiency

This appendix provides a detailed explanation of the FreqFold technique, illustrates its operation with a concrete example, and formally connects its benefits to a general principle of Principal Component Analysis (PCA) concerning structured data. This justification clarifies FreqFold's role in minimizing transformation loss towards decoupled RoPE within the RoRoPE framework (Section 4.2).

C.1 Detailed Explanation of FreqFold and RoRoPE's PCA

In the RoRoPE framework, Rotary Position Embedding (RoPE) is applied. RoPE encodes positional information by rotating pairs of feature dimensions. For each RoPE frequency index $l \in \{1, \ldots, d/2\}$, the corresponding pair of dimensions $([2l-1 :: d], [2l :: d])$ from query and key vectors are rotated. When multiple original attention heads are used (say, g heads), and their key/query projection outputs are concatenated, the RoPE operation for a specific frequency index l applies to a $2g$-dimensional vector segment (formed by concatenating the l-th 2D RoPE subspace from each of the g heads). RoRoPE then applies PCA via matrices $\{\mathbf{U}_l\}_{l=1}^{d/2}$ to these $2g$-dimensional segments, independently for each frequency index l.

The core idea of FreqFold is to approximate numerically similar RoPE base frequencies as being effectively identical. For instance, if RoPE uses original base frequencies $\theta_{l_1}, \theta_{l_2}, \ldots, \theta_{l_M}$ that are close in value, MD-FreqFold might treat them all as a single, representative frequency θ^*.

This approximation has a significant implication for how PCA is applied in RoRoPE:

- **Without FreqFold (Standard RoRoPE PCA):** For each distinct RoPE frequency index l, a separate PCA transformation $\mathbf{U}_l$ is learned and applied to the corresponding $2g$-dimensional key/query segments.
- **With FreqFold:** If M original RoPE frequency indices (say $l_1, \ldots, l_M$) are grouped together by FreqFold due to their frequency similarity, the M corresponding $2g$-dimensional segments are effectively concatenated. Instead of M separate PCAs on $2g$-dimensional vectors, a single PCA is performed on the resulting $M \cdot 2g$-dimensional vectors.

C.1.1 Illustrative Example of FreqFold

Let's consider a scenario with $g = 2$ key heads, and each head has $d_{head} = 8$ dimensions. Thus, there are $d/2 = 8/2 = 4$ distinct RoPE frequency indices per head, which we denote as $\phi_1, \phi_2, \phi_3, \phi_4$. The total number of dimensions is $2 \times 8 = 16$. The RoPE angles for these 16 dimensions could be conceptualized as follows (repeating for each pair, and across heads):

- **Head 1 (dims 1-8):** $(\phi_1, \phi_1), (\phi_2, \phi_2), (\phi_3, \phi_3), (\phi_4, \phi_4)$
- **Head 2 (dims 9-16):** $(\phi_1, \phi_1), (\phi_2, \phi_2), (\phi_3, \phi_3), (\phi_4, \phi_4)$

Case 1: RoRoPE without FreqFold For each frequency index ϕ_l, RoRoPE groups the corresponding dimensions from all $g = 2$ heads. Each such group forms $2g = 2 \times 2 = 4$-dimensional vectors (across N samples).

- Group for ϕ_1: Dimensions $\{1, 2\}$ from Head 1 and $\{9, 10\}$ from Head 2. PCA is applied to these N samples of 4D vectors.

- Group for ϕ_2: Dimensions $\{3, 4\}$ from Head 1 and $\{11, 12\}$ from Head 2. PCA is applied to these N samples of 4D vectors.
- Group for ϕ_3: Dimensions $\{5, 6\}$ from Head 1 and $\{13, 14\}$ from Head 2. PCA is applied to these N samples of 4D vectors.
- Group for ϕ_4: Dimensions $\{7, 8\}$ from Head 1 and $\{15, 16\}$ from Head 2. PCA is applied to these N samples of 4D vectors.

Here, RoRoPE performs 4 separate PCA operations.

Case 2: RoRoPE with 2D-FreqFold 2D-FreqFold implies we are pairing up original frequencies. Suppose FreqFold approximates $\phi_1 \approx \phi_2$ (calling this effective frequency $\Phi_A = \phi_1$) and $\phi_3 \approx \phi_4$ (calling this $\Phi_B = \phi_3$).

- **Effective Group for** Φ_A: This group now includes all dimensions originally associated with ϕ_1 OR ϕ_2.
  - Original ϕ_1-dimensions: $\{1, 2\}$ from Head 1; $\{9, 10\}$ from Head 2. (Forms a 4D segment S_{ϕ_1})
  - Original ϕ_2-dimensions: $\{3, 4\}$ from Head 1; $\{11, 12\}$ from Head 2. (Forms a 4D segment S_{ϕ_2})

  With FreqFold, these segments S_{ϕ_1} and S_{ϕ_2} are concatenated. PCA is now applied to the N samples of $(4 + 4) = 8$-dimensional vectors formed by $[S_{\phi_1}, S_{\phi_2}]$. Effectively, dimensions $\{1, 2, 3, 4\}$ from Head 1 are combined with $\{9, 10, 11, 12\}$ from Head 2.
- **Effective Group for** Φ_B: Similarly, this group includes dimensions originally for ϕ_3 OR ϕ_4.
  - Original ϕ_3-dimensions: $\{5, 6\}$ from Head 1; $\{13, 14\}$ from Head 2. (Forms S_{ϕ_3})
  - Original ϕ_4-dimensions: $\{7, 8\}$ from Head 1; $\{15, 16\}$ from Head 2. (Forms S_{ϕ_4})

  PCA is applied to the N samples of 8-dimensional vectors formed by $[S_{\phi_3}, S_{\phi_4}]$.

Here, RoRoPE with FreqFold performs 2 PCA operations, but each operates on larger, 8-dimensional vectors which are concatenations of what were previously separate PCA targets.

C.2 Formalizing the Benefit of FreqFold in PCA

The example above illustrates that FreqFold causes a re-grouping and concatenation of data segments prior to PCA. The benefit of this concatenation is explained by the following proposition. It states that performing PCA jointly on these concatenated segments (as FreqFold enables) is more effective at preserving variance (and thus minimizing loss) than the alternative of performing separate PCAs on the original, smaller segments and then notionally combining their outcomes.

Consider one such FreqFold merge: suppose M original RoPE frequency indices $l_1, \ldots, l_M$ are deemed equivalent by FreqFold. Without FreqFold, each l_p would correspond to a dataset X_p (e.g., N samples of $2g$-dimensional key segments). With FreqFold, these M datasets are concatenated into a single larger dataset $X_{merged} = [X_1, X_2, \ldots, X_M]$, and PCA is applied to X_{merged}.

Proposition 2. *Let M distinct groups of key segments $X_1, X_2, \ldots, X_M$ be identified. Each $X_p \in \mathbb{R}^{N \times d'}$ (where $p \in \{1, \ldots, M\}$) consists of N samples of d'-dimensional vectors. Assume data in each X_p is mean-centered. Let $S_p = \frac{1}{N-1} X_p^T X_p \in \mathbb{R}^{d' \times d'}$ be its covariance matrix. FreqFold causes these M groups to be merged for a single PCA operation.*

Define $V_1 = \sum_{p=1}^{M} \lambda_{p,1}$, where $\lambda_{p,1}$ is the largest eigenvalue of S_p. This V_1 represents the sum of variances if each of the M original groups X_p were individually reduced to its single most dominant dimension.

Let $Z = [X_1, X_2, \ldots, X_M] \in \mathbb{R}^{N \times (M \cdot d')}$ be the dataset formed by concatenating the features (columns) of these M groups. Let $S_{concat} = \frac{1}{N-1} Z^T Z \in \mathbb{R}^{(M \cdot d') \times (M \cdot d')}$ be its covariance matrix. Define $V_2 = \sum_{j=1}^{M} \mu_j$, where $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_M$ are the M largest eigenvalues of S_{concat}. This V_2 represents the variance captured if the concatenated data Z is reduced to M dimensions using PCA.

Then, the variance captured by the joint PCA on the FreqFold-merged data (V_2) is greater than or equal to the sum of variances from optimally reducing each original group to one dimension (V_1):

$$V_2 \geq V_1$$

This proposition explains that FreqFold's strategy of enabling PCA over larger, concatenated segments (formed by merging data from RoPE frequencies deemed similar) is mathematically favored for variance preservation compared to separate, more fragmented PCAs.

C.3 Proof of Proposition 2

The objective is to prove that $V_2 \geq V_1$, using the notation from Proposition 2. The proof strategy is to construct a specific M-dimensional subspace for the concatenated data Z. We show that the variance captured by projecting Z onto this particular subspace equals V_1. Since the PCA procedure yielding V_2 finds the optimal M-dimensional subspace maximizing captured variance, V_2 must be at least V_1.

Let $\lambda_{p,1}$ be the largest eigenvalue of S_p (covariance of X_p), and $\boldsymbol{w}_{p,1} \in \mathbb{R}^{d'}$ be its corresponding eigenvector. So, $S_p\boldsymbol{w}_{p,1} = \lambda_{p,1}\boldsymbol{w}_{p,1}$ and $\boldsymbol{w}_{p,1}^T\boldsymbol{w}_{p,1} = 1$. The variance $\lambda_{p,1} = \boldsymbol{w}_{p,1}^T S_p \boldsymbol{w}_{p,1}$. $V_1 = \sum_{p=1}^{M} \lambda_{p,1}$.

For the concatenated data Z, $V_2 = \sum_{j=1}^{M} \mu_j$. By Ky Fan's theorem for matrix eigenvalues:

$$V_2 = \max_{\substack{U \in \mathbb{R}^{(M\cdot d')\times M} \\ U^TU = I_M}} \mathrm{Tr}(U^T S_{concat} U)$$

where U's columns form an orthonormal basis for an M-dimensional subspace of $\mathbb{R}^{M\cdot d'}$.

Construct $U^* = [\boldsymbol{u}_1^*, \ldots, \boldsymbol{u}_M^*] \in \mathbb{R}^{(M\cdot d')\times M}$. For $p \in \{1, \ldots, M\}$, define $\boldsymbol{u}_p^* \in \mathbb{R}^{M\cdot d'}$:

$$\boldsymbol{u}_p^* = \begin{pmatrix} \mathbf{0}_{d'\times 1} \\ \vdots \\ \boldsymbol{w}_{p,1} \quad \text{(as the } p\text{-th block of size } d') \\ \vdots \\ \mathbf{0}_{d'\times 1} \end{pmatrix}$$

The set $\{\boldsymbol{u}_1^*, \ldots, \boldsymbol{u}_M^*\}$ is orthonormal. The variance retained by projecting Z onto the subspace of U^* is:

$$\mathrm{Tr}((U^*)^T S_{concat} U^*) = \sum_{p=1}^{M} (\boldsymbol{u}_p^*)^T S_{concat} \boldsymbol{u}_p^*$$

Let S_{qr} be the (q, r)-th block of S_{concat}, where $S_{qr} = \frac{1}{N-1} X_q^T X_r$. Note $S_{pp} = S_p$. Each term $(\boldsymbol{u}_p^*)^T S_{concat} \boldsymbol{u}_p^* = \boldsymbol{w}_{p,1}^T S_{pp} \boldsymbol{w}_{p,1} = \boldsymbol{w}_{p,1}^T S_p \boldsymbol{w}_{p,1} = \lambda_{p,1}$. So, $\mathrm{Tr}((U^*)^T S_{concat} U^*) = \sum_{p=1}^{M} \lambda_{p,1} = V_1$. Since V_2 is the maximum possible variance:

$$V_2 \geq \mathrm{Tr}((U^*)^T S_{concat} U^*) = V_1$$

Thus, $V_2 \geq V_1$. This proves Proposition 2.

C.4 Discussion on the Trade-off in FreqFold

While Proposition 2 demonstrates a clear benefit of FreqFold in terms of PCA efficiency—specifically, that merging M original frequency groups allows for greater variance preservation when reducing to M dimensions—it is crucial to acknowledge an inherent trade-off. The foundational assumption of FreqFold is the approximation of numerically similar RoPE base frequencies as effectively identical. This approximation, by its very nature, introduces a degree of deviation from the original, precise RoPE formulation.

The extent of this deviation, and thus the potential loss in the fidelity of positional encoding, typically correlates with how aggressively frequencies are grouped. A larger M or a looser criterion for

similarity when grouping frequencies can amplify this approximation error. Consequently, while increasing the dimensionality of vectors undergoing PCA is beneficial from the perspective of PCA variance capture as shown by the proposition, it may simultaneously increase the lossiness of the RoPE approximation itself. Therefore, the practical application of FreqFold requires a careful balancing act. The parameter M (representing the number of original RoPE frequencies treated as one effective frequency for PCA purposes) or the specific grouping strategy for frequencies must be chosen to optimize this trade-off.

D Balancing Key-Value Norms and Low-Rank Approximation

This appendix elaborates on the Key-Value (KV) balancing technique and the subsequent joint low-rank approximation applied to the NoPE (No Positional Encoding) components of the keys and the values, as mentioned in Section 4.3 of the main paper. After the RoRoPE procedure (Section 4.2), the key projection matrix W^K is effectively split into two components: $W^{DK}_{\text{RoPE}} \in \mathbb{R}^{d \times D}$ corresponding to the single head that retains RoPE, and $W^{DK}_{\text{NoPE}} \in \mathbb{R}^{(g-1)d \times D}$ corresponding to the remaining $g-1$ head components that do not use RoPE. The value projection matrix is denoted as $W^{DV} \in \mathbb{R}^{gd \times D}$.

D.1 KV Balancing: Purpose and Formulation

Purpose The primary goal of KV balancing is to ensure that the principal component analysis (PCA), when applied jointly to the NoPE key and value activations, is not disproportionately influenced by components with larger norms. We observed that the activations derived from W^{DK}_{NoPE} (i.e., $\mathbf{k}_{\mathbf{NoPE},t} = W^{DK}_{\text{NoPE}}\mathbf{x}_t$) often have a significantly larger average norm than those from W^{DV} (i.e., $\mathbf{v}_t = W^{DV}\mathbf{x}_t$). Without balancing, PCA would predominantly capture the variance within the NoPE key components, potentially neglecting important variations in the value components.

Formulation To address this imbalance, we introduce a scaling factor α. This factor is computed as the ratio of the expected L2 norms of the NoPE key activations to the value activations, based on a calibration dataset:

$$\alpha = \frac{\mathbb{E}_t[\|W^{DK}_{\text{NoPE}}\mathbf{x}_t\|_2]}{\mathbb{E}_t[\|W^{DV}\mathbf{x}_t\|_2]} \tag{32}$$

where $\mathbf{x}_t \in \mathbb{R}^D$ is the t-th input token.

While the main paper states scaling W^{DK}_{NoPE} by $1/\alpha$ and W^{UK} by α for mathematical equivalence in the model's output, for the purpose of deriving the PCA projection, we effectively use scaled NoPE key activations. That is, the activations used to compute the PCA basis are $\mathbf{k}'_{\text{NoPE},t} = 1/\alpha \cdot W^{DK}_{\text{NoPE}}\mathbf{x}_t$ and $\mathbf{v}_t = W^{DV}\mathbf{x}_t$. This ensures that the PCA process considers features from keys and values on a more equitable footing with respect to their magnitudes. The subsequent low-rank decomposition will then be applied to W^{DK}_{NoPE} and W^{DV}, using the PCA basis derived from these balanced activations.

D.2 Joint Low-Rank Approximation of NoPE Keys and Values using PCA

After determining the scaling factor α, we proceed to compress the projection matrices associated with the NoPE keys (W^{DK}_{NoPE}) and all values (W^{DV}) jointly.

The process is as follows:

1. **Collect Calibrated Activations**: A small calibration dataset (WikiText-2) is used. For each input $\mathbf{x}_t$ from this dataset, we compute the scaled NoPE key activations $\mathbf{k}'_{\text{NoPE},t}$ and the value activations $\mathbf{v}_t$. These are concatenated to form combined activation vectors:

$$\mathbf{c}_{\text{NoPE},t} = \begin{pmatrix} \mathbf{k}'_{\text{NoPE},t} \\ \mathbf{v}_t \end{pmatrix} \in \mathbb{R}^{(2g-1)d} \tag{33}$$

2. **Perform PCA**: PCA is performed on the set of collected combined activation vectors $\{\mathbf{c}_{\text{NoPE},t}\}$. This involves computing the covariance matrix of these vectors and finding its principal components. The eigenvectors (corresponding to the largest eigenvalues) are selected to form the columns of a projection matrix $R_{KV} \in \mathbb{R}^{((2g-1)d) \times r_{kv}}$, where r_{kv} is the reduced rank. This matrix R_{KV} captures the directions of highest variance in the (balanced) combined NoPE key and value activation space.

Table 2: Composition of the training dataset.

Dataset	Sampling Weight
`fineweb-edu-dedup`	0.70
`cosmopedia-v2`	0.15
`python-edu`	0.06
`open-web-math`	0.08
`stackoverflow`	0.01

3. **Low-Rank Decomposition of Projection Matrices**: Let $W^{DKV} = \begin{pmatrix} W^{DK}_{\text{NoPE}} \\ W^{DV} \end{pmatrix} \in \mathbb{R}^{((2g-1)d)\times D}$ be the initial projection matrix that transforms the input $\mathbf{x}_t$ into an intermediate NoPE Key and Value representation $\mathbf{c}_{\text{NoPE},t} = W^{DKV}\mathbf{x}_t$. Further, let $W^{UKV} = \begin{pmatrix} W^{UK}_{\text{NoPE}} & 0 \\ 0 & W^{UV} \end{pmatrix} \in \mathbb{R}^{2hd\times((2g-1)d)}$ represent the subsequent collective projection matrix that takes $\mathbf{c}_{\text{NoPE},t}$ and processes it to produce the actual keys and values required by the attention mechanism for the NoPE components, where $W^{UK}_{\text{RoPE}} \in \mathbb{R}^{hd\times gd}$ and $W^{UK}_{\text{NoPE}} \in \mathbb{R}^{hd\times(g-1)d}$ are two parts of W^{UK} hat participate in and do not participate in the RoPE computation, respectively. The original sequence of operations for these components can be expressed as $W^{UKV}W^{DKV}\mathbf{x}_t \in \mathbb{R}^{2hd}$, in which the first hd elements correspond to the keys and the following hd elements correspond to the values.

To introduce a low-rank bottleneck, we modify both W^{DKV} and W^{UKV} using the PCA projection matrix R_{KV}.

- The initial projection matrix W^{DKV} is transformed into $W^{DKV'} \in \mathbb{R}^{r_{kv}\times D}$:

$$W^{DKV'} = R^T_{KV}W^{DKV} \tag{34}$$

  This new matrix $W^{DKV'}$ takes the original input $\mathbf{x}_t$ and projects it into a compressed r_{kv}-dimensional latent space, which is the actual content stored in the KV cache for the NoPE components.

- The subsequent projection matrix W^{UKV} is transformed into $W^{UKV'} \in \mathbb{R}^{2hd\times r_{kv}}$:

$$W^{UKV'} = W^{UKV}R_{KV} \tag{35}$$

  This new matrix $W^{UKV'}$ now takes the compressed latent representation as input and produces the final representations for the NoPE components that are used in the attention calculation. As we can see, $W^{UKV'}$ is actually the concatenated form of W^{UK} and W^{UV} in MLA:

$$W^{UKV'} = \begin{pmatrix} W^{UK} \\ W^{UV} \end{pmatrix} \tag{36}$$

This joint decomposition allows for a more holistic compression by identifying shared latent structures between NoPE keys and values, guided by the balanced PCA.

E Experimental Settings of Fine-tuning

Datasets Following the experimental setups of MHA2MLA, we fine-tune our models using the prtraining corpus from SmolLM Ben Allal et al. [2024]. The dataset comprises FineWeb-Edu-Dedup Lozhkov et al. [2024a], Cosmopedia-v2 — a synthetic dataset generated by Mixtral Wu et al. [2024], Python-Edu from StarCoder Lozhkov et al. [2024b], Open-Web-Math Paster et al. [2023], and data from StackOverflow Stack Overflow [2025]. To ensure a fair comparison with the MHA2MLA baseline, we constructed our training dataset using the same data composition strategy. Specifically, we replicate the dataset mixing ratios used in the MHA2MLA setup to maintain experimental consistency, which is shown in Table 2.

Hyperparameters The fine-tuning hyperparameters for models of all sizes are listed in Table 3. In the table, entries with a slash (/) indicate a two-step training process.

Table 3: Training details across different models.

	SmolLM 1B7		LLaMA2 7B		
	-68.75%	**-87.50%**	**-68.75%**	**-87.50%**	**-92.97%**
Batch size	64	64	64	64 / 64	256 / 64
Learning rate	1e-4	1e-4	2e-5	2e-5 / 2e-5	1e-4 / 2e-5
Tokens	300M	1B	500M	2B / 1B	5B / 1B
Warmup ratio	0.03	0.08	0	0 / 0.03	0 / 0.03
lr scheduler	constant	constant	constant	constant / cosine	constant / cosine
Sequence length	2048	2048	4096	4096	4096

F Detail Information for vLLM Benchmark

In Section 5.4, we demonstrated the speedup achieved by TransMLA—which compresses 92.97% of the KV cache—compared to the original LLaMA-2-7B model. This section provides a detailed analysis of throughput across various hardware configurations.

To account for the effects of both the prefilling and decoding stages, we adopt a setting where the input and output lengths are equal. For instance, with a total context length of 1k, we set the input length to 512 tokens and the output length to 512 tokens. Most experiments are conducted using 100 requests to compute the average throughput. However, for shorter context lengths such as 1k, inference is extremely fast, leading to some timing fluctuations. To mitigate this, we increase the number of requests to 1000 for more stable measurements.

While the original LLaMA-2-7B model supports a maximum context length of 4096 tokens, we extend this limit to 32k tokens in our evaluation. Detailed throughput results are presented in Table 4.

On a GPU with 165.2 TFLOPS of compute and 24GB of memory, the LLaMA-2-7B model runs out of memory when the context length reaches 16k tokens. In contrast, TransMLA sustains a throughput of 414.41 tokens per second under the same conditions. On a more powerful GPU with 320 TFLOPS and 64GB of memory, we employ a development version of the vLLM framework. We anticipate that the throughput of TransMLA will improve further with the release of future optimized versions of the framework tailored for this hardware.

Table 4: Throughput comparison between LLaMA-2-7b and TransMLA at varying input lengths and number of requests.

Context Length	Requests	Model	Throughput(output tokens/s)		
			165.2 TF|24GB	312 TF|40GB	320 TF|64GB
1K	1000	LLaMA-2-7b	653.81	1579.26	1249.13
		TransMLA	**3043.65**	**4062.43**	**1798.17**
2K	100	LLaMA-2-7b	352.85	850.14	789.31
		TransMLA	**2241.87**	**2577.01**	**1080.73**
4K	100	LLaMA-2-7b	173.09	441.37	442.63
		TransMLA	**1318.78**	**1926.15**	**1021.03**
8K	100	LLaMA-2-7b	85.80	218.51	216.66
		TransMLA	**832.69**	**1118.18**	**870.15**
16K	100	LLaMA-2-7b	OOM	110.58	112.13
		TransMLA	**414.41**	**601.36**	**483.22**
32K	100	LLaMA-2-7b	OOM	38.32	55.69
		TransMLA	OOM	**243.81**	**278.09**

G Case Study

To provide an intuitive understanding of TransMLA's impact on model performance, this section presents several examples from vLLM's docs. We compare the outputs of three model variants: (1) a model with 92.97% of its KV cache compressed without any fine-tuning; (2) a model pretrained on

6B tokens, as detailed in Table 1; and (3) a model fine-tuned for one epoch on the SmolTalk dataset, following the setup described in Allal et al. [2025a]. The results are summarized in Table 5.

As shown in Table 5, even without any additional training, the compressed model is still able to produce coherent and meaningful responses. This demonstrates the effectiveness of techniques such as RoRoPE, FreqFold, and BKV-PCA in significantly mitigating performance degradation. Moreover, with a modest amount of pretraining or supervised fine-tuning (SFT), the model's performance improves substantially. These findings highlight TransMLA's potential as a general framework for converting various GQA models into MLA models, with promising prospects for aligning with the performance of advanced systems like DeepSeek R1.

Table 5: Examples from different model configurations. Red indicates input; black indicates output. "w/o Training" denotes the TransMLA-compressed model (92.97% KV cache) without further training. "Pre-Training" and "Fine-Tuning" show outputs after pretraining on a 6B-token corpus and SFT on SmolTalk Allal et al. [2025b], respectively.

Model	Prompt & Generated Text
w/o Training	Hello, my name is Katiu, my father's dog, the pet of the 3600 year-old tribe, Kint. The Kangs were part of a race of reptiles. A small handful
Pre-Training	Hello, my name is Sasha and I am in third grade at Meadows. You may be wondering what this article is about. Well, I have been doing a lot of research on the water cycle and decided to write about it.
Fine-Tuning	Hello, my name is Emily, and I'm a 20-year-old college student. My hobbies include painting, writing, and photography. I also enjoy playing the guitar.
w/o Training	The president of the United States is elected by the legislature. The legislature controls the national armed forces, but only provides the funds to establishing a national guard.
Pre-Training	The president of the United States is elected to a four-year term by the people of each state in a general election held every four years on the Tuesday following the first Monday in November.
Fine-Tuning	The president of the United States is not a position to be taken lightly. This person is the chief executive of the United States of America, and has immense power and influence.
w/o Training	The capital of France is Paris. Its geographical position in the Iberian Plain of France, Spain, Spain, and Morocco are the four largest cities. This region is located in Asia, Spain and Morocco.
Pre-Training	The capital of France is a major business city and it is a favorite destination for businesses from all over the world. It has a strategic location in the heart of the European Union, which makes it one of the most popular cities in Europe.
Fine-Tuning	The capital of France is Paris, and it is one of the most popular tourist destinations in the world. It is a city that offers something for everyone, from art and history to food and fashion.
w/o Training	The future of AI is in serious risk to create a major breakthrough in this emerging phenomenon in the history of artificial intelligence.
Pre-Training	The future of AI is looking bright. With advancements in technology and the increasing availability of data, AI is expected to become more intelligent and capable of performing even more complex tasks.
Fine-Tuning	The future of AI is The future of AI is more nuanced and complex than we might think. Here are some potential developments that could shape the future of AI.