Multi-matrix Factorization Attention

Anonymous ACL submission

Abstract

We propose novel attention architectures, Multi-matrix Factorization Attention (MFA) and MFA-Key-Reuse (MFA-KR). Existing variants for standard Multi-Head Attention (MHA), including SOTA methods like MLA, fail to maintain as strong performance under stringent Key-Value cache (KV cache) constraints. MFA enhances model capacity by efficiently scaling up both the number and dimension of attention heads through low-rank matrix factorization in the Query-Key (QK) circuit. Extending MFA, MFA-KR further reduces memory requirements by repurposing the key cache as value through value projection re-parameterization. MFA's design enables strong model capacity when working under tight KV cache budget, while MFA-KR is suitable for even harsher KV cache limits with minor performance trade-off. Notably, in our extensive and large-scale experiments, the proposed architecture outperforms MLA and performs comparably to MHA, while reducing KV cache usage by up to 56% and 93.7%, respectively.

1 Introduction

The decoder-only transformer with standard Multi-Head Attention (MHA) (Vaswani et al., 2017; Radford, 2018) has become the de facto architecture for large language models. Its autoregressive nature enables the reuse of cached attention key-value tensors (KV cache) from previous tokens, significantly relieving the computation overhead during the step-by-step decoding (Pope et al., 2023). However, the KV cache memory footprint scales linearly with both batch size and sequence length, leading to large amount of memory occupancy and traffic, which becomes the primary bottleneck during the decoding phase of LLM (Yuan et al., 2024).

To address these challenges, Multi-Query Attention (MQA) and Grouped Query Attention (GQA) reduce KV cache usage by sharing key and value projections across heads (Shazeer, 2019; Ainslie et al., 2023). Similarly, Multi-head Latent Attention (MLA) applies low-rank compression to key and value projections and only caches the latents (DeepSeek-AI et al., 2024). However, all methods fail to match MHA's performance under stringent KV cache budgets (Touvron et al., 2023), as the added constraints on key and value projections limit the capacity of the attention module.

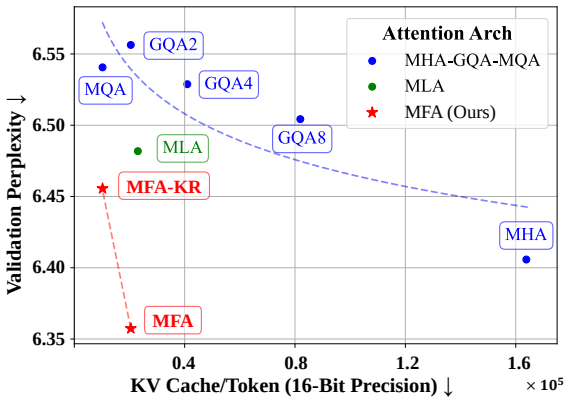

Figure 1: Validation perplexity vs. KV cache memory usage across different attention architectures in a 1B setting. KV Cache/Token indicates the KV cache size in bytes per token, assuming 16-bit precision for each element. Lower is better for both axes.

Driven by these limitations, we analyze the modeling capacity in attention mechanisms and present a unified perspective on existing MHA variants. Our analysis reveals that the number and dimension of attention heads are critical for maintaining modeling capacity—an under-explored design aspect in current methods (Dubey et al., 2024; Muennighoff et al., 2024; Jiang et al., 2024). This insight highlights the need to scale these factors efficiently to mitigate the capacity degradation caused by existing KV cache-saving techniques, pushing attention modules closer to their theoretical upper bound.

Inspired by this understanding, we propose **Multi-matrix Factorization Attention (MFA)**, a novel attention module, along with its variant, **MFA-Key-Reuse (MFA-KR)**. These attention

modules are specifically designed to enhance modeling capacity under strict KV cache constraints.

Specifically, MFA employs a low-rank matrix factorization in the Query-Key (QK) circuit (Elhage et al., 2021), enabling parameter-efficient scaling of both the number and dimension of heads without excessive kv cache usage. Building on MFA, MFA-KR reuses the key cache as value through a re-parameterized value projection with original key projection and a light weight gated projection. This minor modification cuts KV cache usage by an additional 50% with negligible performance trade-offs. Moreover, methods like MLA add complexity to support widely-adopted position embedding (i.e. RoPE), while our proposed MFA family naturally fit in current LLM training and inference ecosystems, ensuring practical adoption without introducing additional architectural complexity.

We conduct extensive experiments to evaluate the performance and KV cache efficiency of MFA and MFA-KR, alongside detailed ablation studies on their design. Impressively, our proposed attention architecture is the only approach that performs comparably to standard MHA in terms of accuracy while adhering to strict KV cache constraints. Specifically, in a 7B parameter model trained on 1T tokens, MFA and MFA-KR reduce KV cache usage by up to 93.7% while achieving superior or comparable benchmark accuracies compared to MHA.

2 Background: Capacity Analysis of Attention

In order to delimit the scope of our analysis, we introduce the concept of Generalized Multi-Head Attention (GMHA). It encompasses all multi-head mechanisms with linear query-key (QK) and value-output (VO) circuits, and per-head softmax attention. The QK circuit determines how information propagates between entities, and the VO circuit dictates how information is transformed (Elhage et al., 2021). Fundamentally, GMHA can be described and analyzed using inference formulation and factorization formulation. The inference formulation highlights how keys and values are computed and cached during inference, and factorization formulation clarifies the model's capacity by interpreting QK and VO matrices as low-rank factorizations. This offers a unified perspective on how different factorization strategies mediate the trade-off between model capacity and efficiency.

Within this framework, we identify Fully Parameterized Bilinear Attention (FPBA) as the upper bound of capacity. MHA and its variants can be regarded as a low-rank decomposition of FPBA, making FPBA a unified theoretical reference point for analysis. Building on this understanding, we propose general design principles for constructing efficient and effective attention modules. These principles inform the design of the Multi-Matrix Factorization Attention (MFA) mechanism, which is introduced in the next section.

2.1 Fully Parameterized Bilinear Attention

Inspired by the work of (Shazeer et al., 2020), FPBA is defined as follows:

$$O_i = \sum_{c=1}^{H} \Big(\sum_{j=1}^{i} \phi(\frac{x_i W_c x_j}{\sqrt{H}}) x_j U_c \Big), \qquad (1)$$

where ϕ denotes the softmax operator, H is the embedding dimension, and W_c, $U_c \in \mathbb{R}^{H\times H}$ are independently parameterized for each channel c. FPBA adheres to three key design principles to reach the theoretical maximum capacity within the GMHA framework. i. Channel-specific interactions. In FPBA, each channel c has a dedicated parameter W_c, the QK circuit $x_i W_c x_j$ captures channel-specific relations between $\mathbf{x}_i$ and $\mathbf{x}_j$. ii. The additivity of the c-th channels of x_i and x_j is generally not holding true. The VO circuit $x_j U_c$, which is fully parameterized as $U \in \mathbb{R}^{H\times H\times H}$, and enables the projection of the H-dimensional embedding of x_j into arbitrary permutation of the H-dimensional embedding of x_i; iii. Full utilization of representations. FPBA fully utilizes the H-dimensional representations of both $\mathbf{x}_i$ and $\mathbf{x}_j$, without compressing any dimensions. This flexibility allows unrestricted interactions across all dimensions, setting FPBA as the upper bound of capacity within the GMHA framework.

2.2 Analysis of MHA and Its Variants

As the prototypical instance of GMHA, MHA can be expressed using inference formulations (2) and factorization formulations (3), as shown below.

$$O_i = \sum_{c=1}^{n} \Big(\sum_{j=1}^{i} \phi(\frac{x_i Q_c (x_j K_c)^T}{\sqrt{d}}) x_j V_c \Big) O_c^T \quad (2)$$

$$= \sum_{c=1}^{n} \Big(\sum_{j=1}^{i} \phi(\frac{x_i (Q_c K_c^T) x_j^T}{\sqrt{d}}) x_j V_c O_c^T \Big), \quad (3)$$

where $Q_c, K_c, V_c, O_c \in \mathbb{R}^{H\times d}$ represent the query, key, value, and output projections for c-th head. Comparing Eqs. (1) and Eqs. (3), we can see that MHA is mathematically equivalent to a version of FPBA where W_c and U_c are approximated with low-rank factorization $Q_c K_c^T$ and $V_c O_c^T$ separately, both with bottleneck of d. During inference time, by sharing parameters among d channels rather than having a distinct set of parameters for each channel, and given that typically $nh = H$, the KV cache per token is reduced to $2H$.

MQA extends the parameter sharing principles of MHA by using a single set of key and value parameters across all attention heads. The formulations for MQA are nearly identical to those of MHA, with the key difference being that W_c and U_c are factorized into $Q_c K^T$ and $V O_c^T$, where $K, V \in \mathbb{R}^{H\times d}$ are shared among all heads, each retaining rank d but with shared parameter constraints. In inference formulations, by eliminating head-specific K_c and V_c parameters, the KV cache size is decreased to $2d$.

MLA adopts a more complex factorization of FPBA as follows:

$$O_i = \sum_{c=1}^{m} \Big(\sum_{j=1}^{i} \phi\Big(\frac{x_i S_q Q_c (x_j S_k K_c)^T}{\sqrt{d}}\Big) x_j S_v V_c \Big) O_c^T \quad (4)$$

$$= \sum_{c=1}^{m} \Big(\sum_{j=1}^{i} \phi\Big(\frac{x_i (S_q Q_c K_c^T S_k^T) x_j^T}{\sqrt{d}}\Big) x_j S_v V_c O_c^T \Big), \quad (5)$$

where $S_q, S_k, S_v \in \mathbb{R}^{H\times C}$ are shared among all heads, $Q_c, K_c, V_c \in \mathbb{R}^{C\times d}$ are head-specific parameters, and C denotes the dimensionality of the latent factorization. We omit the decoupled RoPE design here for simplicity. Comparing factorization formulations Eq. (5) with Eq. (1), it becomes evident that MLA employs parameter sharing across every H/m channels. Specifically, W_c is decomposed $S_q Q_c K_c^T S_k^T$, and U_c is decomposed $S_v V_c O_c^T$. Although the intermediate dimension $C > d$ typically, the overall rank remains to be the smallest dimension as d, without promoting the expressive capacity of the model.

3 Multi-matrix Factorization Attention

Building upon the analysis in the previous section, we arrive at the general design objective for efficient and effective attention module: to find a matrix factorization scheme that minimizes parameter and KV cache size while pushing the model's capacity as close as possible to that of FPBA.

Following these principles, we introduce Multi-matrix Factorization Attention (MFA), incorporating three key design strategies: (1) Increasing the number and dimension of heads to minimize the amount of channel sharing in the propagation process and to provide greater expressive freedom for each head; (2) Applying aggressive low-rank matrix factorizations on W^n to enhance parameter efficiency as the model scales; (3) Utilizing single-key-and-value-head techniques to maintain minimal KV cache usage.

The inference and factorization expressions of MFA are given by:

$$O_i = \sum_{c=1}^{n} \Big(\sum_{j=1}^{i} \phi\Big(\frac{x_i S_q Q_c (x_j S_k)^T}{\sqrt{d}}\Big) x_j S_v \Big) O_c^T \quad (6)$$

$$= \sum_{c=1}^{n} \Big(\sum_{j=1}^{i} \phi\Big(\frac{x_i (S_q Q_c S_k^T) x_j^T}{\sqrt{d}}\Big) x_j S_v O_c^T \Big), \quad (7)$$

where $S_q, S_k, S_v \in \mathbb{R}^{H\times C}$ are shared across heads, $Q_c, O_c \in \mathbb{R}^{C\times C}$ are head-specific projection, and C denotes the low-rank factorization dimension.

During inference, as shown in Eq. (6), the key and value for each token x_j are calculated as $x_j S_k$ and $x_j S_v$ respectively, reducing the KV cache per token to $2C$. Compared to FPBA, the weight matrix W_c is decomposed into $S_q Q_c S_k^T$, and the transformation matrix U_c is decomposed into $S_v O_c^T$, both maintaining a rank of C. This decomposition offers several advantages: (1) **Scalable Head Count**: MFA allows for an increase in the number of heads with minimal parameter overhead ($\approx CH$ additional parameters per extra head). Moreover, the KV cache size remains constant regardless of the number of heads; (2) **Enhanced Head Expressiveness**: each head in MFA has a rank of $C > d$ of others typically. This higher rank improves the expressive capacity of each head, allowing for more nuanced propagation and transmission; (3) **Compatibility with Positional Encodings**: unlike MLA, MFA seamlessly integrates with mainstream positional encodings such as Rotary Positional Encoding (RoPE), ensuring broader applicability across various transformer architectures.

To further optimize KV cache usage under stringent memory constraints, we introduce an extension of MFA called MFA-Key-Reuse (MFA-KR). This variant reuses the key cache by re-parameterizing the value projection based on the key projection, effectively reducing the KV cache size by an additional 50%. The re-parameterization

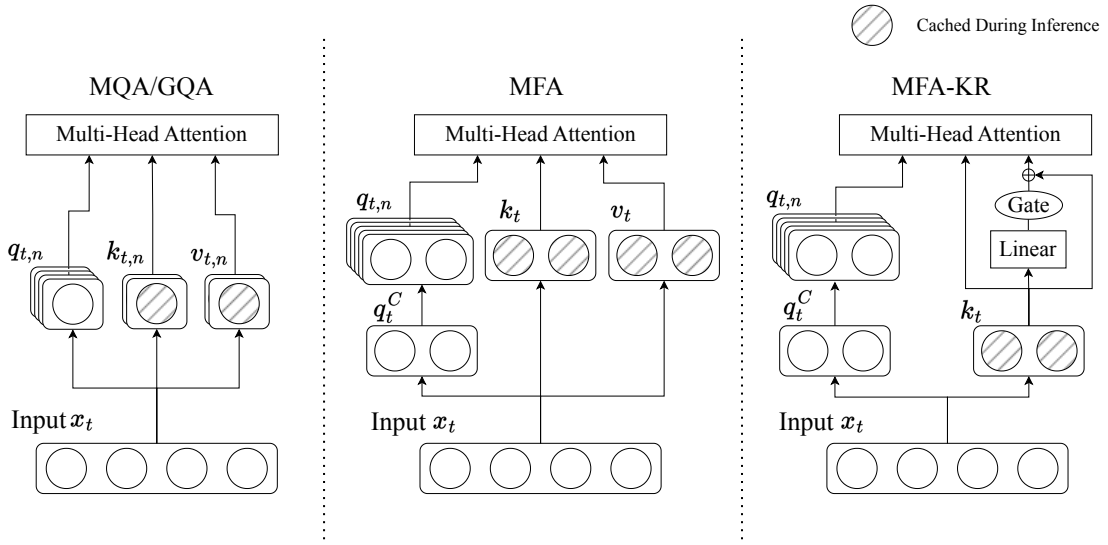

Figure 2: Simplified illustration of MFA/MFA-KR architecture compared with MQA/GQA architecture. By the expanding both the number and dimension of heads in a single-key-and-value manner, MFA/MFA-KR significantly enhances the model capacity under strict KV cache budget while maintaining parameter efficiency during scaling.

is defined as:

$$S_v = S_k + \alpha \odot N S_k \tag{8}$$
$$= (I + \mathrm{diag}(\alpha) N)\, W_K, \tag{9}$$

where $N \in \mathbb{R}^{C \times C}$, $\alpha \in \mathbb{R}^{C}$, and $\odot$ denotes element-wise multiplication. During training, the parameter α is initialized as a zero vector to ensure that S_v equals S_k when training begins, because we empirically found it crucial for maintaining training stability.

To clarify the differences of MFA to other architectures, we present a straight-forward comparison in Table 1. To maintain clarity, we omit MFA-KR and jointly key-value compressed version of MLA. A GMHA model's capacity is influenced by two primary factors: Total Effective Rank (TER) and Shared Latent Subspace Dimension (SLSD). TER is defined as the product of the number of heads and the factorization rank per head (FRH), with higher TER indicating greater overall capacity. On the other hand, SLSD represents the dimension of the latent space shared across all heads. A smaller SLSD reduces the KV cache size but constrains the model's capacity. It is essential to note that the FRH must not exceed the SLSD, establishing a critical trade-off between capacity and efficiency.

As shown in Table 1, MFA achieves a higher TER compared to other methods, positioning it as the closest approximation to the theoretical upper-bound capacity represented by FPBA. Specifically, i. Comparison with MQA: MFA achieves both a higher SLSD and a higher TER; ii. Comparison with MLA: under similar parameter budgets, MFA achieves a smaller KV cache size, a higher TER, and an equivalent SLSD; iii. Comparison with MHA: while MFA has a smaller SLSD than MHA, its TER is higher, leading to empirically superior results as shown in next section.

4 Experiments

We evaluate MFA for large language models from the following perspectives. First, we compare MFA and MFA-KR to MHA at 7B-scale MoE models with 1T training tokens on benchmark accuracies and KV cache usage. Second, we present the loss and KV cache curves of MFA and MFA-KR on increasing training scales. Third, we conduct extensive comparison with existing architecture variants and demonstrate the advantage of MFA and MFA-KR. Finally, we present studies on various design choices and validate the compatibility with different position embeddings

4.1 Common Experimental Settings

In all our experiments, we train our models with language modeling loss on a high-quality training data corpus created internally, including web text, mathematical material, and code, tokenized using the BPE (Sennrich, 2015) tokenizer with vocabulary size of 65536. We adopt pre-normalization using RMSNorm (Zhang and Sennrich, 2019), SwiGLU (Shazeer, 2020) activation function for FFN without dropout, and rotary position embeddings (Su et al., 2024) with base frequency set to 500,000 (Dubey et al., 2024). All models are

Method	KV Cache	Parameter	Heads	Factor. rank per head	Shared latent subspace Dim.	Total effec. rank
FPBA	$2H^2$	$2H^3$	H	H	H	H^2
MHA	$2H$	$4H^2$	n	d	H	nd
MQA	$2d$	$(2+2/n)H^2$	n	d	d	nd
GQA	$2gd$	$(2+2g/n)H^2$	n	d	gd	nd
MLA	$2C+d_r$	$H(3C+d_r+H)$ $+mC(3d+d_r)$	m	d	C	md
MFA	$2C$	$H(3C+mC)+mC^2$	m	C	C	mC

Table 1: Comparison of KV cache usage, parameter count, and total capacity, highlighting their capacity–efficiency trade-offs. *Factor. rank per head* reflects each head's factorization rank; *Shared latent subspace Dim.* indicates a common projection dimension across head's factorizations; *Total effective rank* summing or combining ranks across all heads as total capacity approximates. Generally, $H > C > d = H/n$ and $m > n$. MFA achieves a higher *Total Effective Rank* compared to other variants, making it the closest capacity approximation to FPBA.

	MHA	**MFA-KR**	**MFA**
# Activated Params	1.2B	1.2B	1.2B
# Total Params	6.9B	6.9B	6.9B
KV Cache/Token ↓	196.6K	**12.3K**	24.6K
BBH (Suzgun et al., 2022)	35.9	34.4	**37.8**
MMLU (Hendrycks et al., 2020)	45.2	43.5	**45.5**
Hellaswag (Zellers et al., 2019)	**68.6**	67.5	**68.6**
WG (Sakaguchi et al., 2021)	60.2	**62.0**	60.7
BoolQ (Clark et al., 2019)	66.0	63.4	**66.2**
PIQA (Bisk et al., 2020)	76.0	75.5	**77.0**
SIQA (Sap et al., 2019)	45.7	45.2	**47.9**
SciQ (Welbl et al., 2017)	71.6	68.8	**74.3**
OBQA (Mihaylov et al., 2018)	37.2	36.0	**38.8**
Ruler (Hsieh et al., 2024)	60.9	60.9	**61.7**
DS1000 (Lai et al., 2022)	11.2	11.0	**12.1**
Math (Hendrycks et al., 2021)	9.1	8.1	**9.4**
Average Acc. ↑	49.0	48.0	**49.9**

Table 2: Benchmark accuracy and KV cache usage comparison among MFA, MFA-KR and MHA baseline. We scale the 7B model to 1 trillion training tokens, and MFA generally outperform MHA while using only 12.5% of KV cache per token. MFA-KR demonstrates even less KV cache usage while compromising performance minimally.

trained from scratch and the weights are initialized in the following method: all weights of linear layers are first initialized from a truncated normal distribution with mean zero and standard deviation 0.02, and then for the output projection of attention and the W_2 of the GLU we divide the initialized value by $\sqrt{2 \cdot \text{layer\_idx}}$, which is adopted from (Narayanan et al., 2021).

All models are trained with AdamW (Loshchilov and Hutter, 2019) optimizer, with β = [0.9, 0.95], eps=10^{-8}, weight decay factor of 0.1 and gradient clipping norm of 1.0. For learning rate schedules, we use a linear warmup for the first 2000 steps and a cosine decay to 10^{-5} for the remainder of training. We set the sequence length to 16384 tokens. We hold out a validation set of ≈ 10M tokens drawn from the same distribution of training data for evaluation purposes.

4.2 Language Modeling Evaluation

We train MoE language models with 7B total parameters and 1B activated parameters on 1T tokens to compare MFA/MFA-KR with the MHA.

Setup. We adopt a modified version of DeepSeek-MoE (Dai et al., 2024) as basic architecture, including shared experts and the first layer using dense FFN, but using coarse-grained experts due to system efficiency considerations. We align hidden size,

layers, the total number and the activated number for parameters across all models. FFN dimensions of the first dense layer, total number and activated number of experts are slightly adjusted to meet the requirements. We employ an expert-level load balance loss (Shazeer et al., 2017), with load balance factor as 0.01. All models are trained with same peak learning rate as 8.4×10^{-4} and the each training batch contains 7.3 million tokens, and the training spans 140K steps, totaling 1 trillion tokens. More details can be found in Appendix A.

For evaluation, we benchmark the models on various downstream tasks within a unified evaluation framework. We select extensive tasks, including reasoning, knowledge, and factual accuracy, providing a holistic assessment of model performance.

Results. Table 2 presents a comparison of MFA, MFA-KR, and MHA on downstream language modeling tasks. The results show that MFA achieves superior average benchmark accuracy (49.9%) compared to MHA (49.0%), while reducing KV cache usage per token by 87.5% (from 196.6KB to 24.6KB). This highlights MFA's ability to balance strong modeling capacity while maintaining exceptionally low KV cache memory usage. MFA-KR further minimizes KV cache usage to just 12.3KB per token—only 6.25% of MHA's storage needs—by reusing key caches as values. While MFA-KR incurs a slight accuracy trade-off, it remains competitive and is well-suited for scenarios where memory constraints are paramount.

4.3 Scalabiliy Experiments

We compare the loss scaling curves between MHA, MFA and MFA-KR. The scaling law is supposed to extrapolate the performance at larger scales.

Setup. We use the same model architecture setup mentioned in Section 4.2. We train MoE language models of various sizes (i.e., 1.0B, 2.1B, 5.5B, 6.9B) and various numbers of tokens (i.e., 10B, 20B, 48B, 69B) while keeping the model sparsity (the ratio of activated number of parameters to total number of parameters) constant. We also add our 7B model with 1T training token experiments in our scaling curve results. We use loss on our valuation set as the evaluation metric. More details are shown in Appendix A.3.

Results. We compare the scalability and efficiency of MFA and MFA-KR with MHA through loss scaling across various model sizes and training tokens, as shown in Figure 3. The top plot shows validation loss curves with respect to ND scale (where N is the number of parameters and D the total training tokens). MFA matches MHA's loss scaling behavior, confirming its strong modeling capacity, while MFA-KR demonstrates a similar trend with a minor performance gap, making it suitable for highly memory-constrained scenarios.

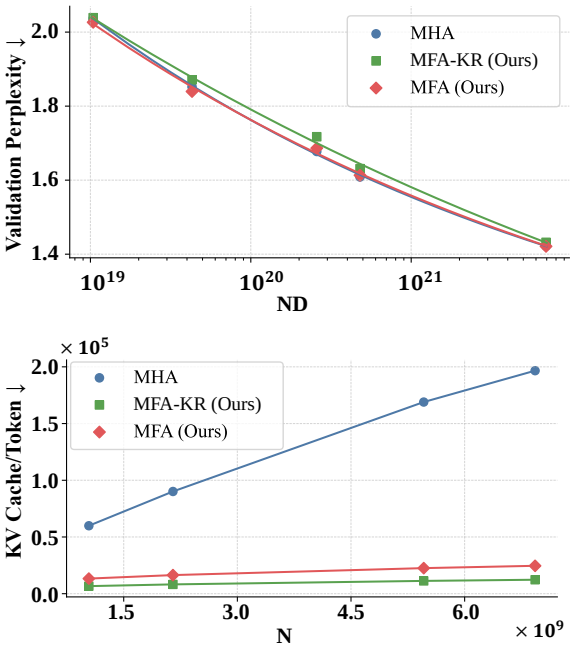

Figure 3: Scaling experiments among MHA, MFA, and MFA-KR. **Top:** Loss vs. ND scale, where N denotes the total number of parameters and D the total training tokens. MFA achieves comparable loss scaling curves to MHA. MFA-KR follows a similar scaling trend, albeit with a slight performance gap. **Bottom:** KV cache usage per token vs. model size. MFA and MFA-KR significantly reduce KV cache usage compared to MHA, with savings growing as model size increases.

The bottom plot compares KV cache usage per token across model sizes. At largest scale, MFA reduces KV cache requirements by 87.5% compared to MHA, with MFA-KR achieving even greater savings at just 6.25% of MHA's usage. The relative savings grow with larger model sizes, highlighting the scalability of both methods.

4.4 Ablation Study

We conduct ablation study on 1B-scale dense model. We set hidden size to 2048, number of layers to 20, and keep the total number of parameter the same by adjusting the FFN size for different attention architectures unless otherwise stated. All models are trained with peak learning rate as 9.63×10^{-4}, and each training batch contains 0.4M

tokens. Total training steps are set to 50k. Therefore, all models consume 20B tokens in training. Ablation studies quantifies the accuracy of models using perplexity evaluated on validation set.

Comparing with Other Attention Architectures. We show the trade-offs between validation perplexity and KV cache usage across various attention architectures in our 1B dense model setting in Figure 1. The comparison includes MHA, GQA, MQA, and MLA, representing a spectrum of design choices for balancing modeling capacity and memory efficiency. Models in the MHA-GQA-MQA spectrum reflect the baseline trade-offs for achievable accuracy and memory usage. More recent MLA architecture is also evaluated in our setting. All models have undergone the same training recipe, except that MLA uses the initialization method mentioned in (DeepSeek-AI et al., 2024). We find that MLA is quite sensitive to the initialization method and only with this initialization can it achieve reasonable performance. Details are elaborated in Appendix B. Our implementation and architecture hyperparameter of MLA refers to the open-source model DeepSeek-V2-Lite.

The results demonstrate that MFA and MFA-KR achieve a new Pareto frontier for accuracy and memory trade-offs. MFA achieves the lowest validation perplexity while using only 12.5% of KV cache memory compared to MHA. MFA-KR further reduces KV cache usage while maintaining competitive accuracy. Notably, MFA and MFA-KR outperform MLA and the MHA-GQA-MQA baselines in terms of both validation perplexity and KV cache efficiency, and MFA achieves even better performance compared to MHA baseline.

Key and Value Projection Design. We conduct a detailed ablation study to evaluate the key and value projection designs in existing works, focusing on key/value sharing introduced by GQA and MQA, as well as key/value low-rank compression proposed by MLA. Results in Table 3 reveal that key compression does not offer performance advantage compared to key sharing under the same KV cache budget and number of model parameters, while introducing additional architectural complexity due to its incompatibility with RoPE. For value projections, our results in Table 3 show that low-rank compression leads to significant performance degradation compared to value sharing under identical KV cache constraints, highlighting that compressing value projections sacrifices more modeling capacity. Based on these findings, MFA adopts key sharing and value sharing, which achieve a favorable balance between simplicity and performance while adhering to strict KV cache budgets.

Efficiently Scale up d and n. To evaluate the parameter efficiency of scaling $d \cdot n$ in MFA, we ablate over QK circuit factorization, as shown in Table 4. First we show that without factorization design, increasing $d \cdot n$ from H to $1.75H$ improves validation perplexity, enhancing the model capacity under strict KV cache usage in this setting. However, vanilla scaling up d and n comes at the cost of higher parameter count (10% more in this setting). In contrast, applying factorization allows MFA to scale $d \cdot n$ to $1.75H$ while keeping the parameter count fixed. This approach achieves the as good validation perplexity, highlighting that the factorization in MFA enables the parameter-efficient scaling of d and n.

Design Choices for Key-Reuse. We ablate the design choice for MFA-KR, as shown in Table 5. Starting from MFA, we incrementally test key reuse and additional design improvements. Vanilla key reusing strategy incur non-negligible performance drop. While adding extra value projection aims to enhance modeling capacity, it suffers from training instability and gets bad performance. Adding a residual connection mitigates instability but still results in suboptimal performance. Finally, incorporating a zero-initialized gating mechanism addresses both stability and performance issues, resulting in MFA-KR, which matches MHA's performance while further halving the KV cache usage compared to MFA.

Different Position Embeddings ALiBi (Press et al., 2021) is also a common position embedding (Almazrouei et al., 2023) with built-in zero-shot length extrapolation ability. Table 6 shows that MFA and MFA-KR maintain advantage with changed position embedding.

5 Related Works

Notable efforts have focused on architectural modifications to minimize KV cache usage besides MQA, GQA and MLA which we elaborated in previous section. CLA(Brandon et al., 2024) and MLKV(Zuhri et al., 2024) attempt to share key and value between layers, further reducing KV cache memory storage overhead. However, since even shared KV cache must be re-loaded in each layers

Config	Extra Op. for RoPE	# Params (B)	Cache/Token (K/V) ↓	Perplexity ↓
Key Projection				
Compressed (k=256)	✓	1.12	12.8K *(K)*	6.36
Shared 2-groups (k=128)	✗	1.06	10.2K *(K)*	**6.32**
Value Projection				
Compressed (v=256)	✗	1.07	10.2K *(V)*	6.60
Shared 2-groups (v=128)	✗	1.06	10.2K *(V)*	**6.32**

Table 3: Comparison between compressed and shared 2-group approaches for key and value projections in attention modules. The column *Cache/Token (K/V)* indicates the size of the key cache (*K*) or value cache (*V*) per token in bytes (16-bit precision). For key projections, the compressed approach (k=256) requires additional operations for RoPE, concatenating another repeated 1-group of key with RoPE along the d dimension. The implementaion is identical to MLA. In key/value projection experiments, the value/key projections use the same multi-head implementations to keep fair comparisons.

Factor.	$d \cdot n$	# Params	C./T. ↓	Val PPL ↓
✗	H	1.08B	20K	6.54
✗	$1.75H$	1.20B	20K	6.38
✓	$1.75H$	1.08B	20K	**6.36**

Table 4: Effect of QK circuit factorization on scaling $d \cdot n$ in MFA. Without factorization, increasing $d \cdot n$ improves validation perplexity increases parameter count. Factorization allows MFA to match validation perplexity while maintaining parameter efficiency, enabling parameter-efficient scaling of d and n. Factor. and C./T. represents factorization and the KV cache/token.

Architecture	KV Cache/Token ↓	Val PPL ↓
MHA	163K	6.41
MFA	20K	6.35
+vanilla KR	10K	6.55
+extra value proj.	10K	7.88
+residual connect	10K	6.65
+gating = MFA-KR	10K	6.45

Table 5: Ablation study for how to arrive at current MFA-KR architecture design choice. KV Cache/Token indicates the KV cache size in bytes per token, assuming 16-bit precision for each element.

seperately, this method does not reduce the KV cache memory traffic, thus having no effect on the latency for core attention computation.

Other works aim to replace all or part of Softmax Attention operations with alternatives that maintain a constant cache state size relative to sequence length, such as SSMs (Gu and Dao, 2024; Lieber et al., 2024) or linear attention(Katharopoulos et al., 2020; Peng et al., 2024). This reduces the cache state size significantly in extremely long-context region, and can be combined with our proposed MFA and MFA-KR in hybrid manner.

Another active area of research seeks to boost the capacity of attention modules. (Bhojanapalli et al., 2020) indentifies the dimension of each head in MHA bottlenecks the capacity of attention module, and the situation may become worse if adhere to current high weight decay training recipe (Kobayashi et al., 2024). Other works like Talking-Head Attention (Shazeer et al., 2020) and DCMHA (Xiao et al., 2024) try to enable information exchange between heads to augment model capacity. Though potential performance gain can be achieved, this modification is not compatible with commonly used Flash Attention (Dao et al., 2022), limiting the scaling up of these architectures.

Architecture	KV Cache/Token ↓	Val PPL ↓
MHA	163K	6.60
MFA-KR	**10K**	6.48
MFA	20K	**6.45**

Table 6: Performance of MFA and MFA-KR compared to MHA with ALiBi as the positional embedding.

Parameter efficiency in transformer models has been extensively studied, especially in finetuning domain (Hu et al., 2021). There are also works focusing on pretraining parameter efficiency like LPA(Lv et al., 2024); however, they do not investigate the effects under limited KV cache budget.

6 Conclusions

We present Multi-matrix Factorization Attention (MFA) and its variant MFA-Key-Reuse (MFA-KR) as scalable solutions to achieve superior performance while drastically reducing KV cache requirements. Our experiments demonstrate that MFA achieves superior benchmark accuracies with up to 87.5% less KV cache compared to MHA, while MFA-KR pushes memory efficiency further by halving KV cache requirements with minimal trade-offs.

Limitations

We do not directly evaluate the system-level implications of KV cache reduction, such as its impact on end-to-end inference efficiency for large-scale, long-context models. The integration of MFA with other architectural innovations, such as CLA or linear attention mechanisms, is not explored. Investigating these combinations could further optimize memory usage and performance, particularly for resource-constrained environments with high model capacity requirements. Moreover, we have not validated the performance of MFA and MFA-KR at even larger scale.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *Preprint*, arXiv:2305.13245.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The Falcon Series of Open Language Models. *Preprint*, arXiv:2311.16867.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. 2020. Low-Rank Bottleneck in Multi-head Attention Models. *Preprint*, arXiv:2002.07028.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. 2024. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. *Preprint*, arXiv:2405.12981.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Preprint*, arXiv:2205.14135.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Hantian Ding, Zijian Wang, Giovanni Paolini, Varun Kumar, Anoop Deoras, Dan Roth, and Stefano Soatto.

2024. Fewer truncations improve language modeling. *arXiv preprint arXiv:2404.10830*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *Preprint*, arXiv:2106.09685.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *Preprint*, arXiv:2401.04088.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *Preprint*, arXiv:2006.16236.

Seijin Kobayashi, Yassir Akram, and Johannes Von Oswald. 2024. Weight decay induces low-rank attention layers. *Preprint*, arXiv:2410.23819.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. *Preprint*, arXiv:2211.11501.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A hybrid transformer-mamba language model. *Preprint*, arXiv:2403.19887.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. *Preprint*, arXiv:1711.05101.

Xingtai Lv, Ning Ding, Kaiyan Zhang, Ermo Hua, Ganqu Cui, and Bowen Zhou. 2024. Scalable Efficient Training of Large Language Models with Low-dimensional Projected Attention. *Preprint*, arXiv:2411.02063.

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

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. 2024. OLMoE: Open Mixture-of-Experts Language Models. *Preprint*, arXiv:2409.02060.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. *Preprint*, arXiv:2104.04473.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. 2024. Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence. *Preprint*, arXiv:2404.05892.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

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

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Rico Sennrich. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Noam Shazeer. 2019. Fast Transformer Decoding: One Write-Head is All You Need. *Preprint*, arXiv:1911.02150.

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

Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou. 2020. Talking-Heads Attention. *Preprint*, arXiv:2003.02436.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *Preprint*, arXiv:1707.06209.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. 2024. Improving Transformers with Dynamically Composable Multi-Head Attention. *Preprint*, arXiv:2405.08553.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

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

A Details of Hyper-Parameters

In this section, we provide more elaboration on the implementation details for Section 4.

A.1 Common experimental Settings

The training data we use in our experiments has gone through thorough cleaning procedure, minimizing the harmful contents and personal information about private individuals. Data is sampled using Best-Fit-Packing (Ding et al., 2024) with bin size of 8 to mitigate truncation issues without

disturbing the data distribution. Samplings are conducted with fixed random seed to ensure fairness when we compare different model architectures. We conduct preliminary experiments to test the validation perplexity fluctuation under same training and evaluation procedure, and find that the standard deviation of validation perplexity is smaller than 0.005. Therefore all our experiments only conduct the training once and apply the standard evaluation protocols. We perform all experiments using PyTorch (Ansel et al., 2024), and the usage adheres to the PyTorch License.

A.2 Language Modeling Evaluation

We present the model hyperparameter for 7B MoE models used in language model evaluation experiment in Table 7.

Architecture	MHA	MFA	MFA-KR
# params (B)	6.9	6.9	6.9
# act. params (B)	1.2	1.2	1.2
Hidden Size	2048	2048	2048
Layers	24	24	24
n	16	18	18
d	128	256	256
# Experts	33	29	29
MoE Top-k	2	2	2
MoE FFN Size	1312	1504	1536
Share FFN Size	2624	3008	3016

Table 7: Architectural hyperparameters for language model evaluation experiment.

A.3 Scalability Experiments

We present model and training hyperparameter details for scalibility experiments.

Exp. Settings	1B	2B	5B	7B
# params (B)	1.0	2.2	5.5	6.9
# act. params (B)	0.2	0.4	0.9	1.2
Train Tokens(B)	10	20	47	69
Hidden Size	1152	1408	1920	2048
Layers	13	16	22	24
n	9	11	15	16
d	128	128	128	128
Learning Rate	8.0e-4	5.9e-4	4.0e-04	3.7e-4
Batch Size (M)	0.3	0.4	1.6	0.8

Table 8: Common hyperparameters for scaibility experiments at each scaling setting.

A.4 Ablation Study

We present the detailed model architecture hyperparameters used in our ablation study, including Feed-Forward Network (FFN) size, the number of attention heads n, and the head dimension d, as shown in Table 9. The low rank dimension for MLA is set to 512, and the dimention with RoPE are set to 64, following DeepSeek-V2-Lite.

Architecture	FFN Size	n	d
MHA	6008	16	128
GQA8	6680	16	128
GQA4	7032	16	128
GQA2	7200	16	128
MQA	7304	16	128
MLA	6504	16	128
MFA	7168	14	256
MFA-KR	7232	14	256

Table 9: Model architecture hyperparameters for the ablation study. The table includes Feed-Forward Network (FFN) sizes, the number of attention heads (n), and head dimensions (d) for different attention architectures.

B Initialization Study on MLA

In our experiments, we find MLA is very sensitive to the initialization method, performing poorly under our default setting. While MHA and MFA remain robust across different initializations. We leave the investigation for the underlying reasons as interesting future work. The experimental results are summarized in Table 10.

Architecture	Initialization	Val PPL ↓
MLA	Ours	6.73
	DeepSeek	6.48
MHA	Ours	6.41
	DeepSeek	6.44
MFA	Ours	6.36
	DeepSeek	6.43

Table 10: Validation perplexity (Val PPL) of MLA, MHA, and MFA under different initialization methods. MLA shows significant sensitivity to initialization, with a large performance gap between our default setting and DeepSeek's method. In contrast, MHA and MFA exhibit robust performance across both initialization settings.

C Potential Risks

Although we conduct detailed processing to filter harmful content, the pretrain models we study can still generate harmful or biased content due to its unaligned nature.