Align Attention Heads Before Merging Them: An Effective Way for Converting MHA to GQA

Qingyun Jin[1,2] **Xiaohui Song**[2] **Feng Zhou**[2] **Zengchang Qin**[1,3] *

[1]School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

[2]OPPO AI Center, Beijing, China

[3]College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam

{jinqingyun, zcqin}@buaa.edu.cn

{songxiaohui, zhoufeng1}@oppo.com

Abstract

Large language models (LLMs) have demonstrated exceptional performance across diverse natural language processing tasks. However, as the model size and the input sequence's length increase, the linearly increasing key-value (KV) cache significantly degrades inference throughput. Therefore, grouped-query attention (GQA), as an alternative to multi-head attention (MHA), has been widely introduced into LLMs. In this work, we propose a cost-effective method for converting MHA into GQA with any compression ratio of KV heads. The key point of our method lies in the application of Procrustes analysis to the attention heads, which enhances the similarity among attention heads while preserving computational invariance, thereby improving the model's post-training performance. Subsequently, we employ L_0 regularization to prune redundant parameters. The model after pruning can be adapted to the standard GQA framework. Experimental results show that our strategy can compress up to 87.5% KV heads of LLaMA2-7B model and 75% KV heads of Sheared-LLaMA-1.3B with acceptable performance degradation. Our code is released at https://github.com/fpcsong/mha2gqa.

1 Introduction

Recently, large language models (LLMs) (Radford et al., 2018; Brown et al., 2020; Ouyang et al., 2022) show remarkable performance on a variety of natural language processing tasks. However, since most LLMs are based on Transformer architecture (Vaswani et al., 2017), the expansion of the sequence length during inference leads to a linear increase in the memory footprint of key-value (KV) cache, which substantially increases on-device memory consumption. A reduced KV cache footprint not only lowers inference costs but also facilitates the processing of longer sequences and improves inference speed. Therefore, reducing the size of the KV cache is a key issue for LLMs.

Many methods for KV cache compression have been proposed, including KV cache quantization (Hooper et al., 2024; Yue et al., 2024; Yang et al., 2024b), tokens dropping (Adnan et al., 2024; Liu et al., 2023a; Tang et al., 2024) and so on. However, these approaches often introduce additional computational procedures, which are incompatible with general LLM frameworks.

Another approach to KV cache compression is to directly change the attention architecture. Multi-query attention (MQA) (Shazeer, 2019) and grouped-query attention (GQA) (Ainslie et al., 2023) reduce KV cache by allowing multiple attention heads to share a single KV head, offering a simple and effective solution for attention optimization. Since GQA has better inference stability and performance, it has been widely used in LLaMA 2 (Touvron et al., 2023), LLaMA 3 (Dubey et al., 2024), Qwen2 (Yang et al., 2024a), Mistral (Jiang et al., 2023) and other LLMs (Liu et al., 2024c; Zhang et al., 2024). Multi-head latent attention (MLA) (Liu et al., 2024a) further reduces KV cache through low-rank projection of the cached data. MLA has been used in the DeepSeek model series (Liu et al., 2024a,b). These efficient attention architectures achieve KV cache compression with better universality.

In this study, we present a method that converts MHA to GQA to compress KV cache. Inspired by the idea of computational invariance in LLMs (Ashkboos et al., 2024) and Procrustes analysis (Schönemann, 1966), we apply proper orthogonal transformations to the projection matrices in attention heads to simplify the conversion from MHA to GQA: Specifically, we regroup attention heads based on the similarity of their KV caches and use generalized Procrustes analysis (Wikipedia contributors, 2022) to maximize the similarity of attention heads within each group. This transformation pre-

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 8804–8816

serves the model's output invariance. Finally, L_0 regularization (Louizos et al., 2017) is applied to transfer original KV heads to new ones to get GQA. Figure 1 illustrates the overall framework of our method. Experimental results show that attention head transformation can significantly improve the performance of the pruned model. Our contributions are as follows.

- Based on the idea of computational invariance, we employ Procrustes analysis to enhance the similarity among attention heads. This approach not only improves the performance of GQA model, but also provides a new perspective for evaluating the similarity between attention heads, offering new insights for future research related to compressing KV cache.
- We propose a general and cost-effective method for converting MHA to GQA, using L_0 regularization to compress the key and value heads to any percentage and basically restore performance after supervised fine-tuning.
- We conduct experiments on LLaMA2-7B (Touvron et al., 2023) and Sheared-LLaMA-1.3B (Xia et al., 2023), converting them into GQA models of varying sizes, separately. Model performance does not decrease significantly compared to that of MHA model.

2 Related Works

2.1 L_0 regularization

L_0 regularization (Louizos et al., 2017) is a structured pruning approach that transforms a pruning problem into an optimization problem under constraints. The pruning process is performed simultaneously with model optimization by introducing trainable masks. With the wide application of LLMs, this method has been applied to compressing LLMs. In the work of (Wang et al., 2019), the L_0 method is applied based on low-rank pruning to achieve further improvements in effectiveness, and they propose to gradually increase the target size at a linear rate during the process of pruning training. In CoFi (Xia et al., 2022), the L_0 method is applied directly to LLMs by introducing pruning masks with different granularities. They prune the hidden dimension, the intermediate dimension, the number of attention heads, and even an entire MHA or FFN layer. The subsequent work Sheared-LLaMA (Xia et al., 2023) incorporates previous methods and specifies the target structure so that the pruned model can be directly adapted to standard LLM frameworks.

2.2 Transfer MHA to GQA

(Ainslie et al., 2023) proposes GQA for the first time, in which MHA is converted to GQA using mean pooling initialization. However, this method requires uptraining to restore performance and incurs significant computational costs. (Yu et al., 2024) keeps the corresponding parameters based on the principal components of the collected KV caches, then uses LoRA (Hu et al., 2021) to fine-tune the model to restore performance. (Chen et al., 2024) proposes to regroup attention heads based on the criterion of cosine similarity and allows for varying group sizes. DHA(Chen et al., 2024a) adaptively configures group sharing for key heads and value heads across various layers, achieving a better balance between performance and efficiency. However, none of the aforementioned improvement methods can be fully adapted to the standard GQA model.

2.3 Compressing model based on the principal components of features

Some previous works (Liu et al., 2023b; Yu and Wu, 2023) have pointed out that the features of LLMs are generally low-rank. Therefore, identifying and deleting the low-rank components of the model is an effective method for model compression.

Low Rank BERT (Noach and Goldberg, 2020) reduces the number of parameters and increases inference speed by decomposing the weight matrices into two low-rank matrices. SliceGPT (Ashkboos et al., 2024) introduces the idea of computational invariance in Transformer architecture and removes columns or rows of the transformed weight matrices to reduce model size. (Yu et al., 2024) applies orthogonal transformations to key-value projection matrices by analyzing the low-rank features of KV cache.

3 Method

In this section, we will specifically describe our method. Our method consists of two parts, transformation of attention heads and pruning training. Transformation of attention heads represents employing Procrustes analysis to align projection matrices in order to enhance the similarity between

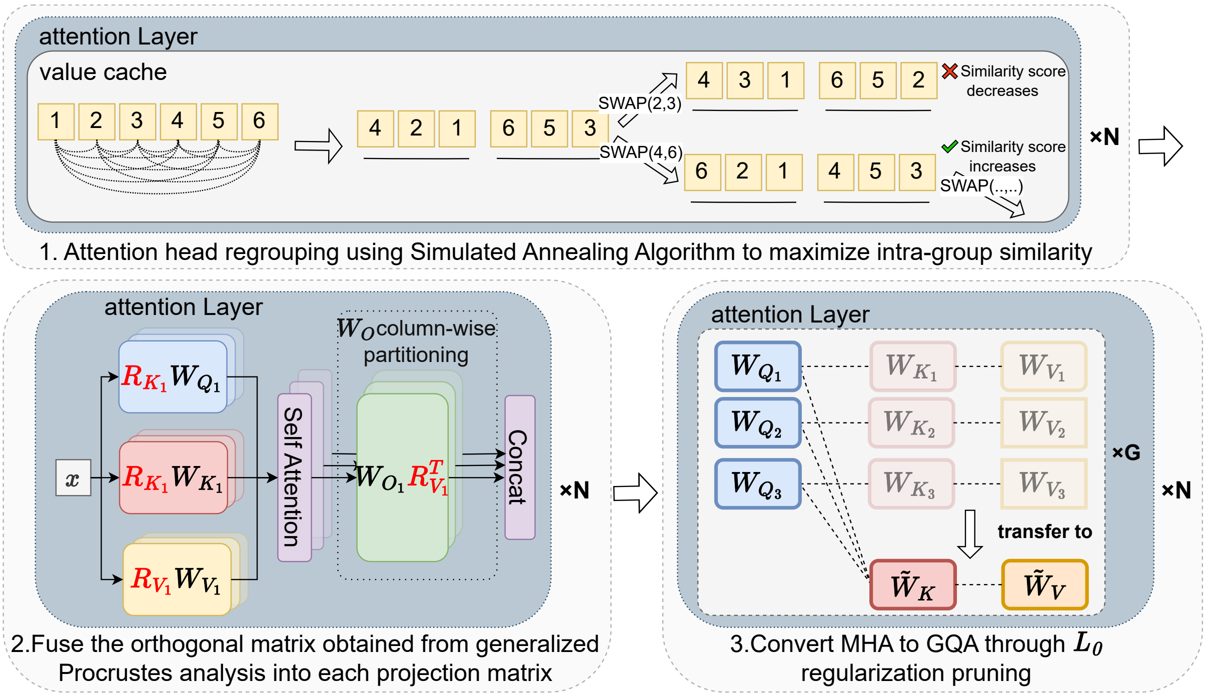

Figure 1: An illustration of our method. 1. In each attention layer of LLM, after the average similarity score is evaluated by Procrustes analysis between every two value caches (or key caches). Use Simulated Annealing Algorithm to search the optimal grouping result that maximizes the similarity score among value caches (or key caches) in each layer. 2. After grouping, then fuse the orthogonal matrix obtained from generalized Procrustes analysis into each projection matrix to enhance the similarity among attention heads in each group without changing the model. 3. During training, L_0 loss is used to gradually transfer original key-value projection matrices to newly added ones within each group (there are G groups in one attention layer). After pruning, original key-value projection matrices will be discarded, then we get a standard GQA model.

attention heads of the same group, so that we can increase efficiency of merging attention heads. The pruning training process combines pruning with L_0 regularization (Louizos et al., 2017) and knowledge distillation (Gou et al., 2021).

3.1 Motivation

To analyze the characteristics of KV cache, we follow a prior calibration method for LLMs (Frantar and Alistarh, 2023; Sun et al., 2023) in order to obtain calibration data: Sample 128 sequences from the C4 (Raffel et al., 2020) training set and each sequence is 2048 tokens long, 262144 tokens in total. Then perform model inference on LLaMA2-7B and collect KV caches, i.e.,

$$K = [K_1; \ldots; K_H] \qquad V = [V_1; \ldots; V_H] \quad (1)$$

where $K, V \in \mathbb{R}^{d \times N}$ are KV caches corresponding to each block, which can be divided into $K_i, V_i \in \mathbb{R}^{d_H \times N}$, N is the number of tokens, d is embedding dimension and H represents the number of heads in each MHA, d_H is set to d/H, then we can calculate the average cosine similarity between each of two heads as follows:

$$SimK_{i,j}^{ori} = \frac{1}{N}\sum_{n=0}^{N-1} cos(K_i[n] \cdot K_j[n]) \quad (2)$$

$$SimV_{i,j}^{ori} = \frac{1}{N}\sum_{n=0}^{N-1} cos(V_i[n] \cdot V_j[n]) \quad (3)$$

where i, j are any two of attention heads in the same block, n represents the n^{th} token in this cache. Taking LLaMA2-7B as an example, as shown in Figure 2, we notice that the vast majority of KV caches are almost orthogonal.

However, according to (Yu et al., 2024), KV caches are low-rank. Given that these caches occupy only a portion of spatial dimensions, applying appropriate orthogonal transformations to the projection matrices to align key and value caches can increase efficiency of merging attention heads. Fortunately, this approach is feasible.

3.2 Preliminaries

Given two sets of vectors of the same shape: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{M \times N}$ and $\mathbf{Y} =$

$\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{M\times N}$, how to find the optimal orthogonal transformation that aligns the two sets of vectors? This kind of problems is called Orthogonal Procrustes problem, and its mathematical formulation is as follows:

$$\min_{\mathbf{Q}} \|\mathbf{Q}\mathbf{X} - \mathbf{Y}\|_F^2 \tag{4}$$

The optimal orthogonal transformation can be derived from SVD of the matrix $\mathbf{Y}\mathbf{X}^T$, the general solution is (Schönemann, 1966):

Perform SVD on the covariance matrix of $\mathbf{X}$ and $\mathbf{Y}$,

$$(\mathbf{Y}\mathbf{X}^T) = \mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Psi}^T \tag{5}$$

Then obtain the optimal orthogonal matrix $\mathbf{Q}$:

$$\mathbf{Q} = \mathbf{\Psi}\mathbf{\Phi}^T \tag{6}$$

We can use the same way to align any two KV caches from different attention heads in the same block. Furthermore, if we want to align more than two sets of caches, generalized Procrustes analysis (Wikipedia contributors, 2022) is a good solution. The detailed description is shown in algorithm 1.

Algorithm 1 Generalized Procrustes Analysis

Require: Matrices $X_1, X_2, \ldots, X_H$
Ensure: Aligned matrices $Y_1, Y_2, \ldots, Y_H$
  Initialize $Y_i = X_i$ for all i
  Compute mean shape $\bar{M} = \frac{1}{H}\sum_{i=1}^{H} Y_i$
  repeat
    for $i = 1$ to H **do**
      Compute $\Phi_i \Sigma_i \Psi_i^T = \text{SVD}(Y_i^T M)$
      Update $Y_i = Y_i \Psi_i \Phi_i^T$
    end for
    Update mean shape $\bar{M} = \frac{1}{H}\sum_{i=1}^{H} Y_i$
  until convergence
  return $Y_1, Y_2, \ldots, Y_H$

3.3 Transformation of attention heads

To calculate the optimal orthogonal matrix for each pair of key and value heads, we collect KV caches according to the method mentioned above. Here, we use two criteria to perform Procrustes analysis.

Based on cosine similarity. Firstly normalize each vector in K_i and V_i to roughly reduce the influence of magnitude of the vector:

$$\hat{K}_i[*] = \frac{K_i[*]}{\|K_i[*]\|} \tag{7}$$

$$\hat{V}_i[*] = \frac{V_i[*]}{\|V_i[*]\|} \tag{8}$$

then we can get the optimal orthogonal matrix Q_V to align any two value caches, taking $\hat{V}_i$ and $\hat{V}_j$ as example:

$$(\hat{V}_i \hat{V}_j^T) = \Phi\Sigma\Psi^T \tag{9}$$

$$Q_{V_j} = \Psi\Phi^T \tag{10}$$

For each block, the output of self-attention layer can be seen as the sum of all attention heads:

$$\begin{aligned}&\text{MultiHead}(W_Q, W_K, W_V, W_O)\\ &= \sum_{i=1}^{H}(W_{O_i}(W_{V_i}X)\text{Softmax}\left(\frac{(W_{K_i}X)^T(W_{Q_i}x)}{\sqrt{d_H}}\right))\end{aligned} \tag{11}$$

where the projection matrices in the attention heads are $W_{Q_i}, W_{K_i}, W_{V_i} \in \mathbb{R}^{d_H\times d}$ and $W_{O_i} \in \mathbb{R}^{d\times d_H}$, $X \in \mathbb{R}^{d\times len}$ represents the previous tokens, and $x \in \mathbb{R}^{d\times 1}$ represents the current token. For brevity, Rotary position embedding (RoPE) is ignored here. Then we can fuse the orthogonal matrix into the value projection matrix W_{V_j} and the output projection matrix W_{O_j} to ensure computational invariance:

$$W'_{V_j} = Q_{V_j} W_{V_j} \tag{12}$$

$$W'_{O_j} = W_{O_j} Q_{V_j}^T \tag{13}$$

As for W_Q and W_K, due to the existence of RoPE (Su et al., 2024), Procrustes analysis cannot be applied directly. However, we can divide the d-dimension space into $d/2$ sub-spaces and apply Procrustes analysis in every two dimension just like RoPE, which is to say the orthogonal matrix for key projection matrix should be in this form:

$$R_{K_j} = \begin{pmatrix} R_{\theta_1} & 0 & \cdots & 0 \\ 0 & R_{\theta_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{\theta_{d/2}} \end{pmatrix} \tag{14}$$

where $R_{\theta.}$ is a 2D rotation matrix. Then we fuse the orthogonal matrix R_{K_j} into the query projection matrix W_{Q_j} and key projection matrix W_{K_j}:

$$W'_{Q_j} = R_{k_j} W_{Q_j} \tag{15}$$

$$W'_{K_j} = R_{k_j} W_{K_j} \tag{16}$$

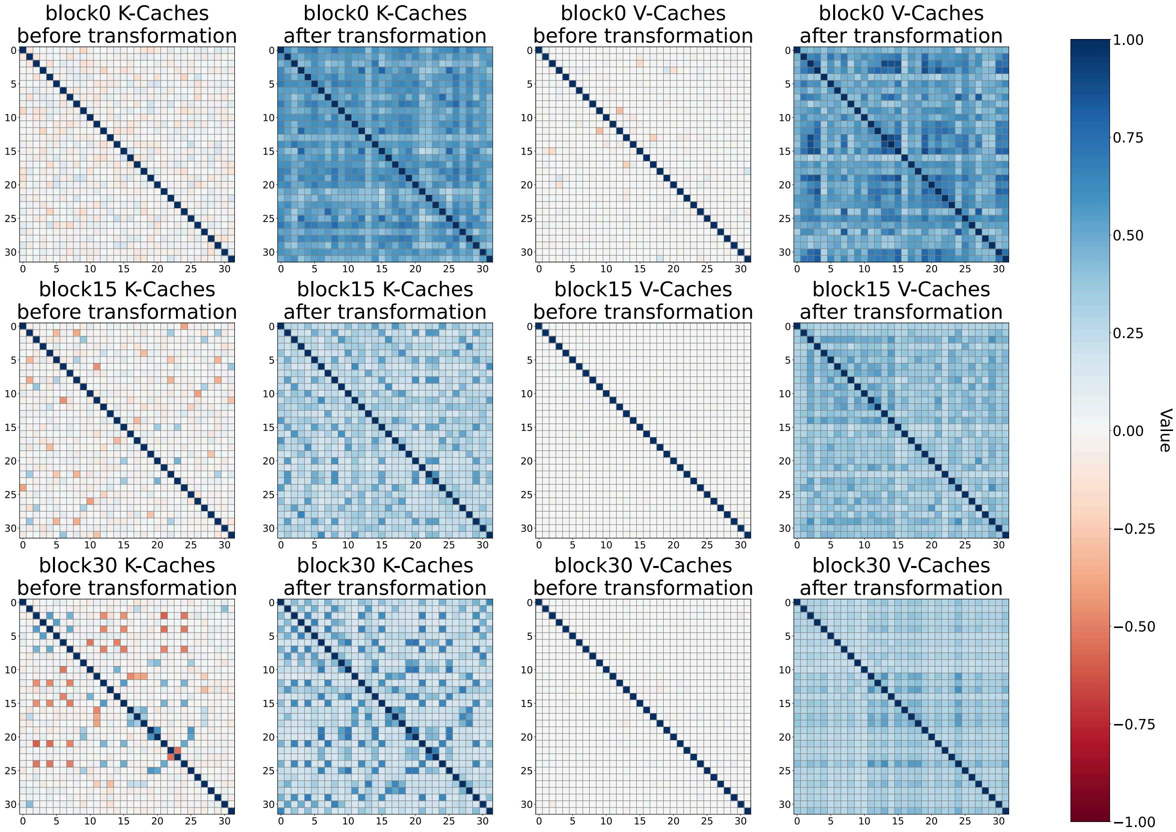

Figure 2: This figure shows the average cosine similarity of key and value caches between any two heads before and after applying transformation in some blocks of LLaMA2-7B. It can significantly improve the cosine similarity between KV caches after Procrustes analysis (In fact, it can also significantly reduce the Euclidean distance between any two caches.).

So, we get

$$\begin{aligned} q_s^{'T} k_t^{'} &= (R^d_{\Theta,s}(R_{K_j} W_{Q_j}) x_s)^T (R^d_{\Theta,t}(R_{K_j} W_{K_j}) x_t) \\ &= x_s^T W_{Q_j}^T (R_{K_j}^T R^d_{\Theta,t-s} R_{K_j}) W_{K_j} x_t \\ &= x_s^T W_{Q_j}^T R^d_{\Theta,t-s} W_{K_j} x_t \\ &= (R^d_{\Theta,s} W_{Q_j} x_s)^T (R^d_{\Theta,t} W_{K_j} x_t) \\ &= q_s^T k_t \end{aligned} \tag{17}$$

where q_s represents the query of the s^{th} position and k_t represents the key of the t^{th} position. This transformation doesn't change the model either.

In this way, given any two key or value caches, we can use this method to calculate the maximum cosine similarity achievable.

$$SimK_{i,j}^{after} = \frac{1}{N} \sum_{n=0}^{N-1} cos(K_i[n] \cdot (R_{K_j} K_j[n])) \tag{18}$$

$$SimV_{i,j}^{after} = \frac{1}{N} \sum_{n=0}^{N-1} cos(V_i[n] \cdot (Q_{V_j} V_j[n])) \tag{19}$$

Noticing $SimV_{i,j}^{after}$ is equal to $SimV_{j,i}^{after}$, so is $SimK^{after}$. Figure 2 shows the cosine similarity between KV caches before and after applying the transformation.

Based on Euclidean distance. Similar to applying transformations based on cosine similarity, we also apply transformations based on Euclidean distance between two KV caches. In this case, we don't normalize vectors and the similarity between two caches can be described as the negative value of the Euclidean distance of them, for brevity, only some key formulas are displayed here:

$$(V_i {V_j}^T) = \Theta \Lambda \Omega^T \tag{20}$$

$$P_{V_j} = \Omega \Theta^T \tag{21}$$

$$SimV_{i,j}^{after} = -\frac{1}{N} \sum_{n=0}^{N-1} \|V_i[n] - (P_{V_j} V_j[n])\|_F \tag{22}$$

In the next section, we will compare the performance of the two criteria.

3.4 Find better grouping method

After obtaining the similarity scores between every pair of attention heads, we can regroup attention heads based on these scores. We define the similarity score of a group as the sum of similarity scores between every pair of attention heads within that group, and the total similarity score for each grouping result is the sum of similarity scores of all groups. Our objective is to identify the grouping result with the highest total score[1]. We use $SimK^{after}$ and $SimV^{after}$ as grouping criteria, respectively. In the next section, we will compare the performance of the two criteria. The mathematical expression of the score of a grouping result $A = \{A_1, A_2, \cdots, A_G\}$ is as follows:

$$Score_{key}(A) = \sum_{g=1}^{G} \sum_{0 \le i < j < D} SimK^{after}_{A_g[i], A_g[j]} \tag{23}$$

$$Score_{value}(A) = \sum_{g=1}^{G} \sum_{0 \le i < j < D} SimV^{after}_{A_g[i], A_g[j]} \tag{24}$$

where A_g is the g^{th} group in G groups, the elements in A_g are the serial number of an attention head and there are $D = H/G$ heads in a group.

We use Simulated Annealing Algorithm to get the best grouping result: Swap two random heads in different groups and calculate the new score, accepting the new result if it reaches a higher score. Repeat this process for multiple iterations. Because initialization has a significant impact on the final result, we run the algorithm multiple times. The details of the algorithm 2 are shown below.

After grouping, we can use Generalized Procrustes analysis to align attention heads in the same group.

3.5 Adaptation of L_0 regularization

During pruning training, we add new projection matrices which are initialized by mean-pooling all the original heads within that group to the model (Ainslie et al., 2023), here we use $\tilde{W}_{K_{b,g}}$ or $\tilde{W}_{V_{b,g}}$ to represent new projection matrices of the g^{th} group in the b^{th} block of the model:

$$\tilde{W}_{K_{b,g}} = \frac{1}{D} \sum_{i=1}^{D} W_{K_{b,(g-1)*D+i}} \tag{25}$$

[1]While the highest similarity between pairs within the same group does not necessarily equate to the lowest cost in terms of converging to the same parameters during pruning, this strategy remains acceptable when considering computation and time costs.

Algorithm 2 Simulated Annealing

Require: $maxIter$, $epoch$, $SimV$ (or $SimK$)
Ensure: grouping result with the highest score $bestG_n$

```
Set score_best ← −∞
for i = 1 to epoch do
  Initialize solution G_n randomly
  score_current ← calculate_score(G_n, SimV)
  if score_current > score_best then
    Set score_best ← score_current
    Set bestG_n ← G_n
  end if
  for j = 1 to maxIter do
    G'_n ← Randomly swap two elements from different groups in G_n
    score_new ← calculate_score(G'_n, SimV)
    if score_new > score_current then
      Set G_n ← G'_n
      score_current ← score_new
      if score_new > score_best then
        Set score_best ← score_new
        Set bestG_n ← G_n
      end if
    end if
  end for
end for
return bestG_n
```

$$\tilde{W}_{V_{b,g}} = \frac{1}{D} \sum_{i=1}^{D} W_{V_{b,(g-1)*D+i}} \tag{26}$$

These new matrices will be trained together with the model and replace original KV heads after pruning. Assume the model has N_{blocks} blocks and H heads in an attention layer, we introduce L_0 masks $z \in \mathbb{R}^{N_{blocks} \times H}$ (Louizos et al., 2017) to achieve this goal:

$$W^{apply}_{K_{b,j}} = z_{b,j} W_{K_{b,j}} + (1 - z_{b,j}) \tilde{W}_{K_{b,g}} \tag{27}$$

$$W^{apply}_{V_{b,j}} = z_{b,j} W_{V_{b,j}} + (1 - z_{b,j}) \tilde{W}_{V_{b,g}} \tag{28}$$

where $g = \lceil \frac{j}{D} \rceil$, $W_{K_{b,j}}$ and $W_{V_{b,j}}$ are the original projection matrices, $\tilde{W}_{K_{b,g}}$ and $\tilde{W}_{V_{b,g}}$ are the newly added projection matrices, $W^{apply}_{K_{b,j}}$ and $W^{apply}_{V_{b,j}}$ are the actual projection matrices employed during pruning. Following the L_0 regularization approach, we parametrize the pruning masks to hard concrete distributions. Initially, each mask is set $z = 1$, we constrain the masks to zero during pruning (Xia et al., 2023). And the original projection matrix

Model	Methods		BoolQ	PIQA	HellaSwag	WinoGrande	ARC-C	ARC-E	OpenbookQA	SIQA	Avg.	Budgets (tokens and epochs)
MHA	Teacher		89.42	77.15	87.62	85.16	73.91	85.09	82.00	78.10	85.48	37.2M 2 epochs
GQA-16	baseline		86.85	81.39	87.08	82.95	66.22	80.88	79.00	77.53	84.60	93M 5 epochs
	default grouping	cos	**88.62**	81.50	89.08	**83.82**	67.56	81.40	**81.40**	77.53	86.08	
		dist	87.98	81.61	89.42	83.14	68.23	**82.63**	79.20	**78.04**	86.16	
	grouping by key	cos	88.47	80.79	88.26	82.64	69.23	82.28	79.20	77.94	85.53	
		dist	88.44	80.25	88.96	82.72	70.23	80.35	77.40	77.28	85.68	
	grouping by value	cos	87.61	81.23	88.39	82.79	69.23	80.35	75.80	77.12	85.28	
		dist	87.80	**82.10**	**89.66**	83.50	**71.24**	81.93	80.80	77.74	**86.35**	
GQA-8	baseline		84.56	79.71	83.53	80.90	60.54	75.26	75.00	76.25	81.65	186M 10 epochs
	default grouping	cos	86.76	80.52	86.66	81.06	64.88	79.30	77.60	**76.92**	84.01	
		dist	86.76	**81.61**	**87.25**	82.64	65.22	**81.05**	77.80	76.61	**84.54**	
	grouping by key	cos	**86.91**	80.68	87.01	82.56	64.21	80.00	76.60	76.20	84.24	
		dist	86.79	80.03	86.39	82.56	65.89	80.00	78.40	76.46	83.94	
	grouping by value	cos	86.39	76.28	81.80	79.64	63.54	74.74	69.20	73.90	80.32	
		dist	86.60	81.50	86.96	**83.74**	**67.22**	79.47	**79.00**	76.31	84.42	
GQA-4	baseline		81.86	76.93	76.97	78.30	55.52	73.86	69.80	74.56	77.03	279M 15 epochs
	default grouping	cos	85.47	78.89	83.18	81.53	59.53	77.02	74.40	75.49	81.53	
		dist	84.83	79.27	83.72	80.74	59.53	77.54	**76.80**	75.54	81.77	
	grouping by key	cos	85.41	79.38	83.37	80.90	61.20	77.19	74.40	75.49	81.66	
		dist	85.26	78.73	81.14	81.21	62.54	75.79	73.20	75.18	80.37	
	grouping by value	cos	**86.18**	79.38	84.05	**82.16**	60.87	76.67	74.00	75.44	82.17	
		dist	85.69	**79.54**	**84.32**	82.00	**63.21**	**77.89**	75.40	**75.64**	**82.36**	

Table 1: Performances of LLaMA2-7B with various methods.

will be transferred to the new matrix when $z = 0$. Unlike traditional L_0 regularization, we aim to eliminate any original key or value heads and just utilize L_0 masks to gradually transfer the original heads to newly added heads. All masks across blocks are constrained by a single loss function:

$$\tilde{\mathcal{L}}_{L_0} = |(\frac{1}{N_{block}H}\sum z) - T| + \left((\frac{1}{N_{block}H}\sum z) - T\right)^2 \quad (29)$$

where T is the target size and equals zero after sparsity warm-up steps.

Before pruning, we already SFT an MHA model as teacher model. Then we use vanilla KL loss and BiLD loss (Li et al., 2024) to encourage alignment of student logits with teacher logits.

$$\tilde{\mathcal{L}}_{distill} = \tilde{\mathcal{L}}_{KL} + \tilde{\mathcal{L}}_{BiLD} \quad (30)$$

To sum up, the overall training loss is:

$$\mathcal{L} = \tilde{\mathcal{L}}_{distill} + \tilde{\mathcal{L}}_{L_0} \quad (31)$$

4 Experiments

4.1 Settings

Model configurations. We apply our method to the LLaMA2-7B model (Touvron et al., 2023) and Sheared-LLaMA-1.3B (Xia et al., 2023) throughout all experiments. We will prune the KV heads of MHA at different pruning rates, and compare them to the fine-tuned MHA model separately.

Datasets. We use the following open-source datasets for training and evaluation: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy (Clark et al., 2018), ARC-challenge(Clark et al., 2018), SIQA (Sap et al., 2019) and OpenbookQA (Mihaylov et al., 2018). The size and instruction template of each dataset are listed in Appendix B.

Implementation Details. We use 1 A100 GPU to perform model transformation, and 8 A100 GPUs for SFT the teacher model and pruning training. We randomly select 128 sequences of 2048 tokens long from the C4 training set (Raffel et al., 2020) as calibration data in attention heads transformation. During the transformation, we convert the model parameters to float64 to reduce the calculation error. The time required for calibration and conversion is detailed in Appendix C. In pruning training, the initial learning rate is 1e-5 for the model parameters and 1e-2 for the pruning masks. The cosine scheduler is employed to reduce the learning rate to 0 by the end of training. More hyperparameter settings can be found in Appendix A.

4.2 Ablation studies

We test the impact of different similarity evaluation criteria (see Section 3.3) and grouping strategies (see Section 3.4). All results are presented in Table 1 and Table 2. Here, "baseline" refers to

Model	Methods		BoolQ	PIQA	HellaSwag	WinoGrande	ARC-C	ARC-E	OpenbookQA	SIQA	Avg.	Budgets (tokens and epochs)
MHA	Teacher		84.83	72.74	68.31	71.35	53.18	64.04	59.20	71.34	71.37	55.8M 3 epochs
GQA-8	baseline		81.99	69.10	58.22	71.51	44.82	61.05	60.20	68.73	64.99	111.6M 6 epochs
	default grouping	cos	83.27	70.95	63.60	71.43	**50.50**	62.63	58.60	70.52	68.38	
		dist	**83.85**	71.11	63.54	71.35	**50.50**	**63.68**	58.60	**71.03**	**68.54**	
	grouping by key	cos	82.72	69.75	62.42	70.96	46.49	61.58	59.60	68.68	67.29	
		dist	82.11	65.34	62.61	69.38	40.80	54.21	49.80	66.22	65.99	
	grouping by value	cos	83.06	70.35	**63.63**	72.14	49.83	61.75	60.40	70.32	68.33	
		dist	83.55	**71.21**	63.54	**73.56**	46.82	61.75	**60.60**	70.57	68.53	
GQA-4	baseline		79.33	64.53	50.42	70.64	38.13	57.02	54.80	66.79	59.55	223.2M 12 epochs
	default grouping	cos	81.41	68.93	59.30	**71.98**	42.14	57.54	56.00	69.04	65.24	
		dist	81.10	**69.20**	59.47	70.80	43.81	59.65	57.20	68.89	65.33	
	grouping by key	cos	81.99	68.23	59.94	71.43	45.15	58.60	59.00	68.83	65.69	
		dist	81.50	67.95	59.54	71.35	**47.83**	57.54	59.80	67.96	65.32	
	grouping by value	cos	81.25	68.28	**60.06**	70.32	47.16	**60.18**	**60.40**	**69.14**	**65.71**	
		dist	**82.29**	67.63	59.77	70.40	46.82	60.00	60.20	**69.14**	65.66	

Table 2: Performances of Sheared-LLaMA-1.3B with various methods.

pruning directly without any transformation, "default grouping" refers to merging adjacent attention heads, "grouping by key" and "grouping by value" indicate grouping attention heads based on key or value cache similarity. "cos" and "dist" represent the transformation based on cosine similarity or Euclidean distance.

4.3 Main results

We report the experimental results and budgets in Table 1 and Table 2, Avg. (Average Accuracy), indicates the average accuracy of all these sub datasets. Except for one set of experiments, all transformed models outperform the baseline. As the sparsity of KV head increases, the advantage of model transformation becomes more obvious, demonstrating the effectiveness of aligning attention heads.

4.4 Analysis of the results

Although our experiments utilize L_0 regularization to accelerate the training process, model transformation can benefit any MHA to GQA conversion process. In certain cases, value-based grouping also contributed to result improvements. For experiments where grouping failed to yield performance gains, we hypothesize that cumulative errors introduced by the transformation may have a negative impact on model performance.

In addition, during the experiments, we found that the model retaining original KV heads failed to converge during the pruning process. That's why we choose not to retain any original KV heads, this setting also allows different pruning speeds for different blocks. Figure 3 shows the actual average size of masks in each block at different target sizes.

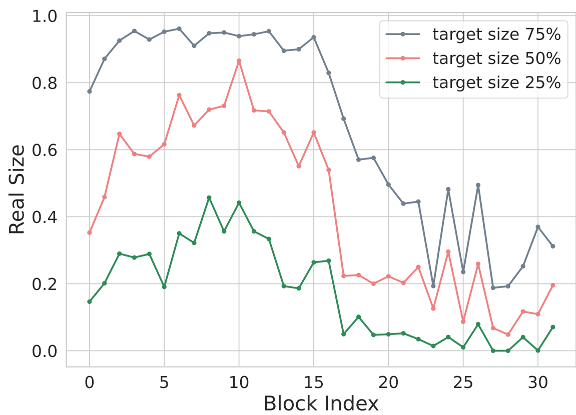

Figure 3: The shared sparsity of L_0 masks across blocks allows different pruning speeds for different blocks in LLaMA2-7B, leading to a more stable training process.

Attention heads closer to the input layer are pruned last, as pruning these heads significantly impacts all subsequent layers.

5 Conclusion

In this paper, we propose a general method for converting an MHA model into a GQA model with any compression ratio of KV heads. We find that applying orthogonal transformations to attention heads based on Procrustes analysis can enhance the similarity between KV heads without changing the model, thereby improving its performance after pruning. Furthermore, we introduce L_0 regularization during pruning training, which reduces the impact of directly eliminating parameters on the model. Our method is applicable to all KV head pruning conditions.

Limitations

Our work has two main limitations. First, we do not delve into the grouping method, and the current approach can be further optimized. Identifying a more effective grouping strategy is one of the potential directions for future research. Moreover, our method entirely relies on the statistical mathematical features of attention heads, without considering their semantic information. In fact, compressing attention heads based on semantic information is also a promising direction (Tang et al., 2024).

References

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6: 114–127, 2024.

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.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

Tony Cai, Jianqing Fan, and Tiefeng Jiang. Distributions of angles in random packing on spheres. *The Journal of Machine Learning Research*, 14(1):1837–1864, 2013.

Yilong Chen, Linhao Zhang, Junyuan Shang, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, and Yu Sun. Dha: Learning decoupled-head attention from transformer checkpoints via adaptive heads fusion. *arXiv preprint arXiv:2406.06567*, 2024a.

Yuang Chen, Cheng Zhang, Xitong Gao, Robert D Mullins, George A Constantinides, and Yiren Zhao. Optimised grouped-query attention mechanism for transformers. *arXiv preprint arXiv:2406.14963*, 2024.

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

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. *arXiv preprint arXiv:1803.05457*, 2018.

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

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

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

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=0LXotew9Du.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Minchong Li, Feng Zhou, and Xiaohui Song. Bild: Bidirectional logits difference loss for large language model distillation. *arXiv preprint arXiv:2406.13555*, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

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*, 2024b.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024c.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023a.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023b.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR, 2023.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Matan Ben Noach and Yoav Goldberg. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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

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

Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31 (1):1–10, 1966.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

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

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads. *arXiv preprint arXiv:2407.15891*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

Wikipedia contributors. Generalized procrustes analysis — Wikipedia, the free encyclopedia, 2022. URL https://en.wikipedia.org/w/index.php?title=Generalized_Procrustes_analysis&oldid=1126373270. [Online; accessed 24-October-2024].

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024b.

Hao Yu and Jianxin Wu. Compressing transformers: features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11007–11015, 2023.

Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm. *arXiv preprint arXiv:2406.07056*, 2024.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

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

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

A Hyperparameter settings

To reduce memory usage, we employ DeepSpeed during both SFT and pruning training, we set k=16 for BiLD loss (Li et al., 2024). During the pruning training process, the sparsity warm-up steps account for 30% of the total steps, during which the target size of the L_0 masks decreases linearly to zero. The maximum pruning steps comprise 80% of the total steps, after which the mask training ceases, only the model parameters are adjusted. Some more hyperparameter settings for SFT teacher model and pruning training are shown in Table 3.

	SFT teacher	pruning training
batch size	128	64
micro batch size	4	1
lr warmup steps	16	32
initial lr of masks	\	1e-2
initial lr of model	1e-5	

Table 3: Some hyperparameters setting for experiments.

B Details of datasets

The sizes of sub datasets are shown in Table 6.

datasets	train	test
BoolQ	9427	3270
PIQA	16113	1838
HellaSwag	39905	10042
WinoGrande	40398	1267
ARC-C	1119	299
ARC-E	2251	570
OpenbookQA	4957	500
SIQA	33410	1954
total	147580	19740

Table 4: Sizes of different datasets

The template of each dataset can be seen in Table 5.

Dataset	Template
Arc-C Arc-E OpenbookQA	Which color shirt will reflect the most light on a hot, sunny day? Choices: ['black', 'blue', 'red', 'white'] Answer:
HellaSwag	Please choose the most appropriate text to complete the passage below: Passage: A male athlete puts powder on his hands. he Choices: ['bends and inspects his hands for damage.', 'shakes them shakingly before putting them in his mouth.', 'mounts a high beam in the gym.', 'then jumps up and does a high jump.'] Answer:
BoolQ	The Coroner – The BBC announced on 2 March 2017 that there would be no further series. Question: will there be a second series of the coroner? Choices: ['true', 'false'] Answer:
Winogrande	Choose the most sensible text to replace the '_' in the following sentence: Natalie was less religous than Patricia, therefore _ attended church services more often on Sundays. Choices: ['Natalie', 'Patricia'] Answer:
PIQA	Goal: how do you flood a room? Choose the most sensible solution to achieve the goal. Choices: ['fill it with objects.', 'fill it with water.'] Answer:
SIQA	Sasha took him to vegas for a vacation. Question: How would Sasha feel afterwards?? Choices: ['sad', 'depressed', 'fulfilled'] Answer:

Table 5: The template of each dataset

C Time cost of model calibration and transformation

model		LLaMA-2-7B	Sheared-llama-1.3B
calibration		5min	3min
GQA16 transformation	w/ grouping	22min	\
	w/o grouping	4min	\
GQA8 transformation	w/ grouping	37min	20min
	w/o grouping	17min	2min
GQA4 transformation	w/ grouping	1h2min	35min
	w/o grouping	29min	14min

Table 6: Time cost of model calibration and transformation. It requires at most one hour to complete this procedure. All calculations are performed by 1 A100 GPU.