Head-wise Shareable Attention for Large Language Models

Zouying Cao[1,2,3], **Yifei Yang**[1,2,3], **Hai Zhao**[1,2,3,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
[3]Shanghai Key Laboratory of Trusted Data Circulation and Governance in Web3
{zuoyingcao,yifeiyang}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Large Language Models (LLMs) suffer from huge number of parameters, which restricts their deployment on edge devices. Weight sharing is one promising solution that encourages weight reuse, effectively reducing memory usage with less performance drop. However, current weight sharing techniques primarily focus on small-scale models like BERT and employ coarse-grained sharing rules, e.g., layer-wise. This becomes limiting given the prevalence of LLMs and sharing an entire layer or block obviously diminishes the flexibility of weight sharing. In this paper, we present a perspective on head-wise shareable attention for large language models. We further propose two memory-efficient methods that share parameters across attention heads, with a specific focus on LLMs. Both of them use the same dynamic strategy to select the shared weight matrices. The first method directly reuses the pre-trained weights without retraining, denoted as **DirectShare**. The second method first post-trains with constraint on weight matrix similarity and then shares, denoted as **PostShare**. Experimental results reveal our head-wise shared models still maintain satisfactory capabilities, demonstrating the feasibility of fine-grained weight sharing applied to LLMs[1].

1 Introduction

Large Language Models (LLMs) have achieved breakthrough performance in a variety of natural language processing tasks (Wei et al., 2022; Bubeck et al., 2023; Zhao et al., 2023). However, such remarkable capability typically comes at the cost of a substantial increase in the model size (Kaplan et al., 2020). Thus, LLMs with billions of parameters (Brown et al., 2020; Touvron et al., 2023) are more resource-hungry despite a wide margin of superiority over small-scale models (Devlin et al., 2019; Liu et al., 2019). This can also pose challenges for deployment on low-capability devices due to limited storage and GPU memory.

To address the high memory requirements of models, weight sharing (Takase and Kiyono, 2021a; Liu et al., 2023) aims to reuse the same parameters to achieve memory- and storage-efficiency while preserving model performance. For small-scale models, e.g., BERT, it is known that several techniques (Lan et al., 2020; Liu et al., 2023) are proposed to explore across-layer parameter sharing. While, Zhang et al. (2022) demonstrate identical weights across different layers are the main cause of training instability and performance degradation. Moreover, the effectiveness of similar techniques at the scale of LLMs remains uncertain.

Thus, we strive to solve this central question: ***Can we design one fine-grained weight sharing strategy that can smoothly apply to large language models*?** For an effective memory-efficient weight sharing method tailored to LLMs, two key challenges must be tackled: a) the choice of shared modules whose weights are reused; b) the trade-off between reducing memory requirements and preserving diverse capabilities.

In the preliminary work, we empirically evaluate the feasibility of weight sharing across the attention heads in LLMs inspired by attention map (i.e., attention scores) reuse. Subsequently, we introduce our design of head-wise shareable attention strategy. It is a simple and intuitive technique for parameter sharing that can be implemented in a few minutes. Specifically, given the pre-trained weight matrices, we concatenate the weight matrix W^q and W^k for each head to measure the cosine similarity that determines which heads can be shared. Meanwhile, head-wise weight sharing promotes parameter diversity in the layers, and thus its performance degradation is acceptable when the

*Corresponding author. This research was supported by the Joint Research Project of Yangtze River Delta Science and Technology Innovation Community (No. 2022CSJGG1400), the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020).

[1]https://github.com/zouyingcao/DirectShare

number of shared parameters is below 30%. Even as weight sharing ratio increases rapidly, our proposed constrained post-training method can narrow the performance drop, which may necessitate additional time.

In summary, our key contributions include:

- We investigate the feasibility of head-wise weight sharing for large language models and propose two corresponding methods named DirectShare and PostShare.
- The proposed DirectShare is time-efficient and retain a large portion of the performance when sharing ratio is below 30%. Complementarily, PostShare yields satisfactory performance via post-training, especially under large ratios.
- Experiments show our proposal achieves comparable performance to the competitive memory-efficient methods. Additional analysis also indicates its efficiency in small-scale models.

2 Related Works

2.1 Memory-efficient Approaches for LLMs

With the growing size of language models, several memory-efficient techniques are proposed to solve. One line to reducing the memory consumption involves network compression, like quantization (Bai et al., 2021; Tao et al., 2022), pruning (Yang et al., 2022; Tao et al., 2023; Yang et al., 2024) and knowledge distillation (Wu et al., 2023; Tan et al., 2023). However, when applied to LLMs, many approaches have become infeasible (Frantar and Alistarh, 2023). To recover performance, they require extensive post-training of the model (Dettmers et al., 2023; Sun et al., 2023).

In addition to these conventional methods, researchers have also investigated more efficient variations of the self-attention mechanism for LLMs (Kitaev et al., 2020; Lv et al., 2023). Reformer (Kitaev et al., 2020) leverages sparsity in the attention layers to improve the efficiency on long sequences and with small memory use. Lightformer (Lv et al., 2023) deploys SVD weight transfer and parameter sharing, which can significantly reduce the parameters on the premise of ensuring model performance. In this paper, our focus is on weight sharing across attention heads.

2.2 Weight Sharing

Weight sharing is a widely used technique (Lan et al., 2020; Liu et al., 2023; Lv et al., 2023; Xu and McAuley, 2023) that aims to improve parameter efficiency and reduce inference memory requirements. Weight sharing enables model compression by eliminating redundant parameters and decouples computation and parameters by reusing the same parameters for multiple computations.

Task-oriented Weight Sharing. One of the prevalent tasks using weight sharing mechanisms is nerual machine translation (NMT). Tied Transformer (Xia et al., 2019) considers model-level sharing and shares the weights of the encoder and decoder of an NMT model. Dabre and Fujita (2019) proposes a method, which shares the weights across all Transformer layers and keeps performance in NMT. Besides, Chi et al. (2021) bring the idea of ALBERT (Lan et al., 2020) to the speech recognition task.

Layer-wise Weight Sharing. Universal Transformer (Dehghani et al., 2019) shares the weights across all layers with a dynamic halting mechanism and improves accuracy on several tasks. Subformer (Reid et al., 2021) utilizes sandwich-style parameter sharing, which only shares the central layers while leaving the first and last layers independent. Takase and Kiyono (2021a) study strategies to explore the best way to prepare parameters of M layers and assign them into N layers (1≤M≤N).

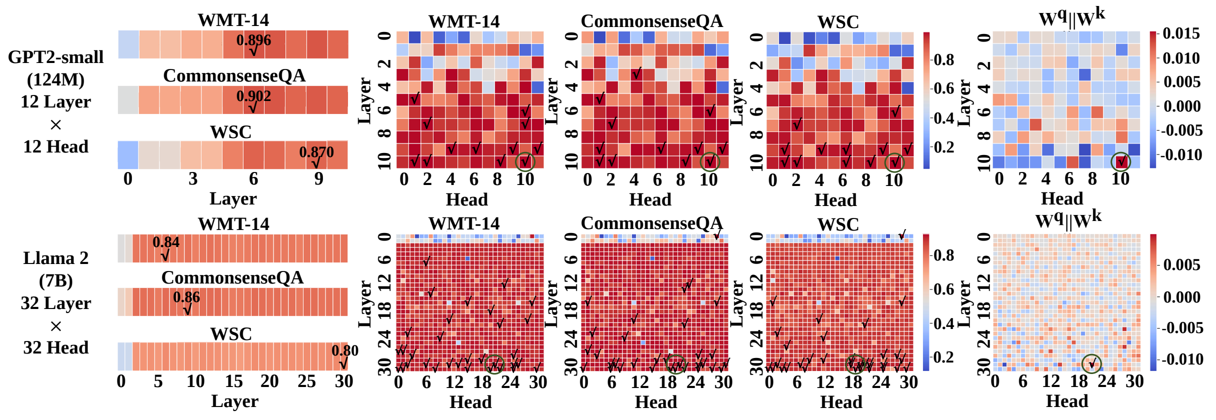
3 Motivation and Empirical Analysis

In this section, we analyze the feasibility of head-wise weight sharing from the perspective of attention map reuse.

3.1 Attention Map Similarity: From Layer-wise to Head-wise

Prior researches (Xiao et al., 2019; Ying et al., 2021; Bhojanapalli et al., 2021) demonstrate the effectiveness of attention map reuse due to the high similarity of attention scores between different layers (especially for adjacency layers). Motivated by this, we delve into attention map similarity, specifically transitioning from layer-wise to head-wise analysis. To measure the evolution of the attention maps over layers and heads, we use the cosine similarity $\mathcal{S}_{cos}$. When $\mathcal{S}_{cos}$ equals one, it means that the attention maps are perfectly similar. Considering two specific self-attention layers, the cosine similarity is calculated as follows:

$$\mathcal{S}_{cos}(\mathbf{A}_p, \mathbf{A}_q) = \frac{\mathbf{A}_p^T \mathbf{A}_q}{\|\mathbf{A}_p\| \|\mathbf{A}_q\|} \quad (1)$$



(a) Layer-wise Attention Map Similarity (b) Head-wise Attention Map Similarity (c) Weight Matrix Similarity

Figure 1: (a) Layer-wise Attention Map Similarity. Taking the last layer as an example, the most similar attention layer with it is marked with $\surd$. (b) Head-wise Attention Map Similarity. $\surd$ mark the top n heads whose attention maps that are most similar to the 6-th head in the last layer(n=the number of heads per layer). (c) Weight Matrix Similarity. ◯ mark the connection between attention map similarity and weight similarity.

where $\mathbf{A}_p, \mathbf{A}_q$ denote the attention map of layers p and q.

We visualize the layer-wise and head-wise attention map similarity across three task-specific datasets: WMT14 (En-Fr) (Callison-Burch et al., 2009), CommonsenseQA (Talmor et al., 2019) and WSC (Levesque et al., 2012). As shown in Fig. 1(a) and (b), the degrees of similarity in attention scores computed in different layers and heads present a certain level of consistency across different tasks. In addition, we find that the cosine similarity values for pairs with high similarity are higher among different heads compared to layers. Specifically, the most similar self-attention layers reach a cosine similarity value of approximately 0.90, while in the case of head-wise comparisons, several pairs have a remarkable similarity of nearly 0.99.

One observation is that as the number of parameters increases, modules with high similarity exhibit variations, particularly in the fine-grained (e.g., head-wise) comparisons within large-scale pre-trained language models. Existing approaches employ "learning to share" techniques to dynamically adjust the sharing strategy (Xiao et al., 2019) or use a uniform sharing strategy but train the modified model from scratch (Ying et al., 2021; Shim et al., 2023). However, such strategies pay little attention to reusing attention map among heads and incur high computational costs for LLMs.

3.2 From Attention Map Similarity to Weight Matrix Similarity

Attention weight matrix similarity provides a complementary perspective to attention map similarity, since the attention scores are calculated based on the weight matrices W^q, W^k. Weight sharing is traditionally based on the assumption that overparameterization is evident in large-scale Transformer models, i.e., the difference in weights decreases as model size increases (Li et al., 2020). In this paper, we explore a potential relationship between attention map similarity and weight similarity.

As mentioned in Section 3.1, head-wise attention map similarity is higher than the cross-layer similarity, while to the best of our knowledge, head-wise attention map reuse is yet to be explored. This might be attributed to the difficulty in finding an optimal dynamic head-wise sharing strategy across different tasks. One intuitive solution is to first measure the attention map similarity between every pair of heads in each dataset separately, and then choose the overlapping modules to share.

Combined with the analysis of weight matrix similarity, we have made a key discovery: given a pre-trained LLM, by concatenating the weight matrix W^q and W^k for each head to measure the cosine similarity, the most similar weight matrix corresponds to the overlapping modules with highly similar attention maps observed across different datasets. As illustrated in Fig. 1(b) and (c), deep green circles mark the connection between attention map similarity and weight similarity (more analysis in Appendix B).

This finding implies that attention heads with high weight matrix similarity also demonstrate analogous attention map similarity regardless of the datasets and model size. Furthermore, since

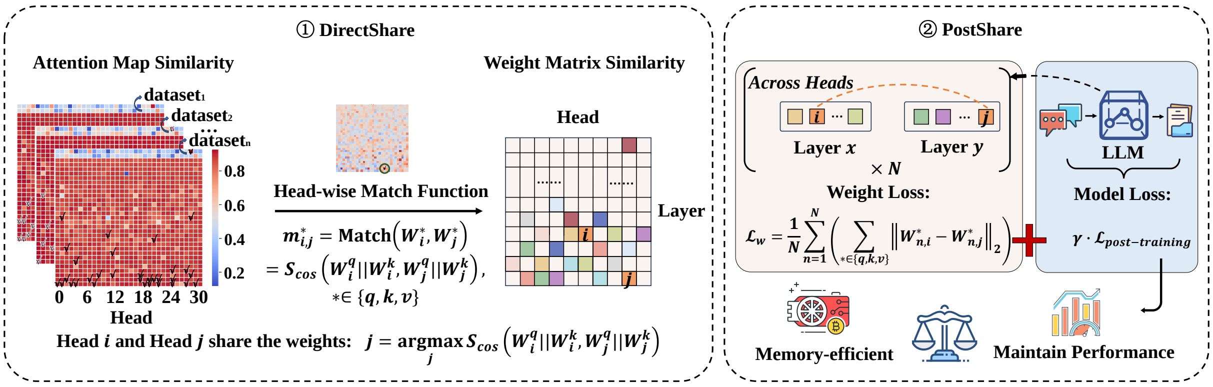

① DirectShare
Attention Map Similarity
dataset₁
dataset₂
datasetₙ
Head
Head-wise Match Function
$m^*_{i,j} = \text{Match}(W^*_i, W^*_j)$
$= S_{cos}(W^q_i||W^k_i, W^q_j||W^k_j)$, $* \in \{q,k,v\}$
Weight Matrix Similarity
Head
Layer
Head i and Head j share the weights: $j = \arg\max_j S_{cos}(W^q_i||W^k_i, W^q_j||W^k_j)$
② PostShare
Across Heads
Layer x
Layer y
$\times N$
Weight Loss:
$\mathcal{L}_w = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{*\in\{q,k,v\}}\|W^*_{n,i} - W^*_{n,j}\|_2\right)$
LLM
Model Loss:
$\gamma \cdot \mathcal{L}_{post-training}$
Memory-efficient
Maintain Performance

Figure 2: ① **DirectShare**: Inspired by attention map reuse, directly share weight matrices across different heads based on cosine similarity; ② **PostShare**: To balance the memory usage and the performance, implement post-training with the constraint of weight matrix similarity and then share.

different heads within the layer present sufficient diversity (Zhou et al., 2021; Vig, 2019), we suppose that weight sharing among these heads can result in higher model behavior consistency compared to layer-wise weight sharing. Thus, we further propose a simple yet effective method for head-wise weight sharing, especially validating its feasibility in large-scale models.

4 Head-wise Shareable Attention

Inspired by Section 3, we present a perspective on head-wise shareable attention for LLMs. Based on one straightforward yet effective weight sharing strategy, we propose two complementary methods, named **DirectShare** and **PostShare**. The overview of our proposal is presented in Figure 2.

4.1 Head-wise Weight Sharing Strategy

Multi-Head Attention (MHA) block is essentially a procedure that computes the relevance of each token in a sentence with respect to all other tokens. Let L be the number of input tokens and M be the number of attention heads in total. Given the input $X \in \mathbb{R}^{L\times D}$, we can obtain queries, keys, and values in the i-th ($1{\le}i{\le}M$) head via three weight matrices, denoted by $W_i^q \in \mathbb{R}^{D\times d_q}$, $W_i^k \in \mathbb{R}^{D\times d_k}$ and $W_i^v \in \mathbb{R}^{D\times d_v}$, respectively. D is the embedding dimension, and $d_q, d_k(= d_q), d_v$ represent the dimensions of three weight matrices, respectively.

To investigate the strategy of weight sharing applied to all the above three weight matrices across heads for LLMs, we perform preliminary experiments in the choice of head-wise match functions **Match**$(\cdot,\cdot)$. For the match functions, inputs are the weight matrices of head $\boldsymbol{i}, \boldsymbol{j}$ and outputs are called matching scores $\boldsymbol{m}$. The higher the score, the more likely it is to share parameters across the heads.

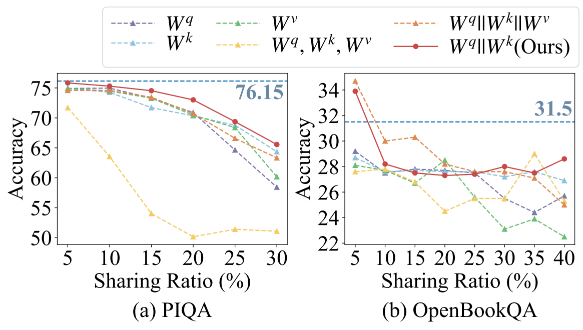


Figure 3: Experiments performed on PIQA and OpenBookQA using different head-wise match functions for Baichuan2-7B model.

$$\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \textbf{Match}(W_i^*, W_j^*), * \subseteq \{q,k,v\} \quad (2)$$

Based on our intuitive analysis in Section 3.2, we choose the cosine similarity between the concatenation matrix of W_i^q and W_i^k:

$$\boldsymbol{m}^q_{\boldsymbol{i},\boldsymbol{j}} = \boldsymbol{m}^k_{\boldsymbol{i},\boldsymbol{j}} = \boldsymbol{m}^v_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^q||W_i^k, W_j^q||W_j^k) \quad (3)$$

Besides, we try another five match functions to compare: (1) Only W_i^q used to measure the cosine similarity, i.e., $\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^q, W_j^q)$; (2) Only W_i^k used to measure the cosine similarity, i.e., $\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^k, W_j^k)$; (3) Only W_i^v used to measure the cosine similarity, i.e., $\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^v, W_j^v)$; (4) Concatenate all the three matrices and then calculate the cosine similarity, i.e., $\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^q||W_i^k||W_i^v, W_j^q||W_j^k||W_j^v)$; (5) Separately use W_i^q, W_i^k, W_i^v to measure the cosine similarity and do weight sharing respectively, i.e., $\boldsymbol{m}^*_{\boldsymbol{i},\boldsymbol{j}} = \mathcal{S}_{cos}(W_i^*, W_j^*)$ and again $* \in \{q,k,v\}$.

Figure 3 shows the results of our exploratory study via DirectShare. As evidenced by the performance curve, using separately weight sharing causes a significant decline in performance compared with sharing the three weight matrices together. And it is enough to do head-wise weight sharing focusing only on the concatenation matrix of W_i^q and W_i^k, since it achieves a favorable trade-off between reducing memory footprint and maintaining performance.

4.2 DirectShare

In practice, we traverse all head pairs to compute matching scores on Equation 3 and for each head, select the one with the highest score to match. When candidate shareable head pairs prepared, we select the top-N pairs in descending order according to the desired sharing ratio α. Finally, we can share the weight matrices together between each selected attention head pairs. A detailed algorithm for our DirectShare is presented in Algorithm 1 and Appendix A.

Algorithm 1: DirectShare using Head-wise Weight Sharing Strategy

```
Input: Sharing ratio $\alpha$, Original LLM $\mathcal{M}$,
Number of layers $\mathcal{L}$,
Number of heads per MHA block $\mathcal{H}$
Output: The LLM $\mathcal{M}^*$ after weight sharing
1  Initialize candidate buffer $\mathcal{D}_\tau$;
2  for $layer_i \leftarrow 2$ to $\mathcal{L}$ do
3      for $i \leftarrow 1$ to $\mathcal{H}$ do
4          $index_i \leftarrow (layer_i, i)$;
5          $index_m \leftarrow$ None;
6          $s_m \leftarrow$ -1;
7          for $layer_j \leftarrow 1$ to $layer_i - 1$ do
8              for $j \leftarrow 1$ to $\mathcal{H}$ do
9                  $index_m \leftarrow (layer_j, j)$;
10                 Compute $\boldsymbol{\mathcal{S}_{cos}}$ using Eq. 3;
11                 if $\boldsymbol{\mathcal{S}_{cos}} > s_m$ then
12                     $s_m \leftarrow \boldsymbol{\mathcal{S}_{cos}}$
13         Store candidate shareable head pair
           $< index_i, index_m, s_m >$ in $\mathcal{D}_\tau$;
14 Sort $\mathcal{D}_\tau$ by descending matching scores $s_m$;
15 $\mathcal{N} \leftarrow$ Top_N($\mathcal{D}_\tau$, $\mathcal{L}$, $\mathcal{H}$, $\alpha$);
16 $\mathcal{M}^* \leftarrow$ Weight_Share($\mathcal{M}$, $\mathcal{N}$).
```

4.3 PostShare

Although DirectShare demonstrates effectiveness in our experiments, we have also encountered noticeable performance drop in minor reading comprehension datasets. To alleviate this problem, we propose PostShare, softly aligning model weights during the post-training process.

With the same sharing strategy (Section 4.1), PostShare first selects the set of weight matrices to share. Next, we incorporate a regularization term into the loss function to constrain our post-training process, encouraging selected weight matrices more similar:

$$\mathcal{L}_w = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} \left(\sum_{*\in\{q,k,v\}} \left\| W_i^* - W_j^* \right\|_2 \right) \tag{4}$$

where $\mathcal{N}$ is the set of selected attention head pairs for sharing. With this regularization weight loss, the proposed PostShare learn model weights by minimizing the following combined loss function:

$$\mathcal{L} = \mathcal{L}_{post-training} + \gamma \times \mathcal{L}_w \tag{5}$$

where $\mathcal{L}_{post-training}$ is the original post-training loss, γ controls the strength of $\mathcal{L}_w$. After the post-training process, the corresponding weight matrices can be shared as DirectShare does. Although post-training indeed increases the time cost of weight sharing, PostShare achieves stable and satisfactory performance across different tasks when reducing memory usage.

5 Experiments

5.1 Experimental Settings

Models. We evaluate DirectShare and PostShare on two open-source LLMs: Llama2 (Touvron et al., 2023) and Baichuan2 (Baichuan, 2023) with 7B and 13B parameters. In PostShare, we use English Wikipedia (Foundation) to post-train the backbone models for weight sharing.

Evaluation. To comprehensively evaluate the model capabilities, we experiment on five distinct tasks: reasoning, understanding, language, knowledge and examination. For consistent comparisons, we deploy open-source LLM evaluation platform OpenCompass (Contributors, 2023).

Baselines. Since existing weight sharing techniques do not support LLMs, we compare DirectShare against Magnitude Pruning (Zhu and Gupta, 2017) and LLM-Pruner (Ma et al., 2023), two influential works for model pruning. Certainly, they are not directly comparable. To ensure fairness in the experiments, both of them only prune the multi-head attention module and thus compare when the same number of parameters is reduced. See Appendix C for additional information.

Benchmark Type		**Reasoning**					**NLU**					**Knowledge**			
Ratio	**Method**	CMNLI	OCNLI	AX-b	AX-g	RTE	RACE-middle	RACE-high	OBQA	CSL	TNEWS	Wino-Grande	BoolQ	C-Eval	MMLU
0%	**Llama2-7B**	32.98	33.12	53.53	55.34	49.82	33.15	35.51	31.80	55.62	20.22	54.04	70.67	32.20	46.69
10%	Magnitude	<u>32.99</u>	30.63	<u>56.70</u>	49.44	47.29	25.42	26.47	**28.20**	49.38	14.85	51.58	60.80	22.16	28.20
	LLM-Pruner	<u>32.99</u>	**33.75**	**57.61**	<u>50.00</u>	<u>48.38</u>	<u>28.20</u>	**30.73**	<u>27.20</u>	<u>53.12</u>	<u>19.76</u>	**52.98**	<u>66.09</u>	<u>22.31</u>	<u>38.11</u>
	DirectShare	**33.00**	<u>32.50</u>	54.17	**51.97**	**50.90**	**28.34**	<u>28.96</u>	**28.20**	**54.37**	**20.86**	<u>52.63</u>	**67.74**	**28.75**	**43.43**
30%	Magnitude	<u>33.16</u>	**35.00**	54.71	50.56	46.93	**21.80**	<u>21.53</u>	25.00	45.62	7.01	**50.88**	44.59	<u>24.38</u>	23.15
	LLM-Pruner	32.99	31.25	<u>56.34</u>	**52.53**	<u>48.74</u>	<u>21.52</u>	**22.21**	**26.80**	<u>50.00</u>	<u>10.20</u>	**50.88**	**54.77**	22.82	<u>25.16</u>
	DirectShare	**33.33**	<u>32.50</u>	**57.07**	<u>51.69</u>	**49.10**	21.45	<u>21.53</u>	<u>26.00</u>	**51.25**	**20.22**	<u>50.18</u>	<u>54.43</u>	**26.24**	**26.53**
0%	**Llama2-13B**	32.99	35.00	58.81	50.56	47.29	60.24	58.03	42.40	58.75	22.13	55.44	71.50	40.17	55.81
10%	Magnitude	<u>32.82</u>	<u>33.12</u>	51.99	**50.56**	**48.38**	22.42	21.78	27.40	51.25	15.39	49.82	62.32	22.52	27.54
	LLM-Pruner	**32.99**	**36.25**	**58.70**	<u>50.00</u>	46.93	<u>51.46</u>	<u>50.80</u>	**47.00**	<u>56.25</u>	**20.95**	**55.44**	<u>68.07</u>	<u>30.25</u>	<u>51.45</u>
	DirectShare	**32.99**	**36.25**	<u>57.61</u>	<u>50.00</u>	<u>47.29</u>	**54.04**	**55.63**	<u>39.40</u>	**56.88**	<u>17.94</u>	<u>54.39</u>	**69.45**	**37.17**	**52.81**
30%	Magnitude	**33.78**	33.75	46.65	<u>50.00</u>	**51.99**	21.80	22.01	**28.80**	46.25	4.19	49.12	56.45	**23.99**	22.86
	LLM-Pruner	<u>32.99</u>	<u>34.38</u>	<u>57.16</u>	**54.21**	45.85	<u>23.96</u>	<u>25.33</u>	26.40	<u>53.75</u>	**16.76**	**51.58**	**63.21**	22.17	<u>27.22</u>
	DirectShare	<u>32.99</u>	**35.00**	**58.33**	<u>50.00</u>	<u>46.57</u>	**26.53**	**27.53**	<u>27.40</u>	**59.38**	<u>16.12</u>	<u>50.18</u>	<u>59.36</u>	<u>22.30</u>	**30.79**

Table 1: Evaluation results of DirectShare based on the Llama2-7B and Llama2-13B models. **Bold** and <u>underline</u> indicate the best and the second best results.

5.2 Main Results

5.2.1 Evaluation on DirectShare

Table 1 shows the overall performance of DirectShare based on Llama2 models. Benchmarks are classified into three categories: reasoning, natural language understanding (NLU) and knowledge-related. The corresponding results for Baichuan2 models can be found in Appendix D.

Logical and Common Sense Reasoning. In the domain of reasoning, when applying a 30% parameter sharing to Llama2-7B, our DirectShare can still maintain an average performance of 99.51% across the five benchmarks, compared to the base model. With the same setting, the shared Llama2-13B retains 99.21% performance. This suggests our finding of head-wise shareable attention for LLMs indeed can work without significant performance degradation in reasoning tasks.

The overall efficacy of our DirectShare rivals with the structured pruning results of LLM-Pruner, without any training. Moreover, our method is quite simple and fast, independent on the original training corpus, while structured pruning will nearly fail in the zero-shot generation tasks without dependencies (Ma et al., 2023).

Natural Language Understanding (NLU). Compared to reasoning tasks, our experimental results unveil a notable performance decrease of approximately 30% in large-scale reading comprehension datasets when applying DirectShare to Llama2-7B model. Beyond this, we discover that on content summary and analysis tasks, DirectShare manages to retain 94.23% of the performance exhibited by the base model. The evaluation results of Llama2-13B align with those of Llama2-7B and we find the accuracy gap is larger as model size increases. This trend also exists in Magnitude Pruning and LLM-Pruner, even the performance drop is larger: LLM-Pruner drops ≈ 3 points more than ours on average while Magnitude Pruning is outperformed by ours by a large margin.

To mitigate this degradation, some post-training pruning methods like SparseGPT (Frantar and Alistarh, 2023) preserves accuracy via the weight update procedure. Similarly, LLM-Pruner uses the low-rank approximation (LoRA, Hu et al., 2022) to post-train the pruned model. Motivated by this, our PostShare proves to be beneficial, substantially improving 17.80% accuracy, albeit at a certain amount of time cost. For more details refer to Section 5.2.2. However, this does not diminish the significance of our DirectShare. The absence of post-training allows us to better understand the feasibility of head-wise weight sharing for LLMs.

Knowledge-related Tasks. As depicted in Table 1,

Ratio	Method	WinoGrande	BoolQ	C-Eval	MMLU	RACE-middle	RACE-high	OBQA	OBQA-fact
0%	**Llama2-7B**	54.04	70.67	32.20	46.69	33.15	35.51	31.8	42.2
30%	DirectShare	50.18	54.43	26.24	26.53	21.45	21.53	26.00	27.60
	PostShare	52.98 ↑ 2.80	66.57 ↑ 12.14	26.38 ↑ 0.14	33.36 ↑ 6.83	29.81 ↑ 8.36	29.45 ↑ 7.92	27.60 ↑ 1.60	33.60 ↑ 6.00

Table 2: Overall Performance of PostShare based on Llama2-7B model. See Appendix E for results on Llama2-13B.

DirectShare takes a clear advantage over other approaches in the field of examination. Our chosen C-Eval and MMLU span diverse disciplines to test both world knowledge and problem solving ability exclusively in a Chinese and English context, respectively. To make this more concrete, Figure 4 vividly contrasts the performance across different subjects based on Llama2-7B on C-Eval and MMLU. But we have to admit directly do weight sharing across attention heads results in a obvious decline in knowledge-related abilities, which can be solved in PostShare.

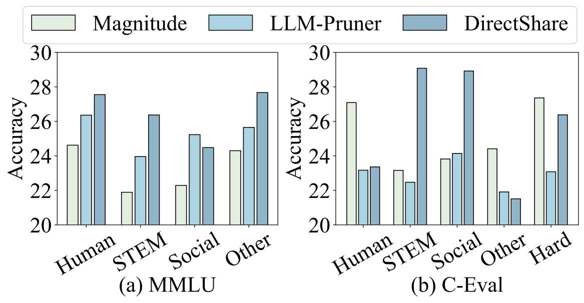


Figure 4: Performance of DirectShare across different subjects based on Llama2-7B on C-Eval and MMLU.

5.2.2 Evaluation on PostShare

Based on the evaluation conducted on DirectShare, we experiment on PostShare, with a special focus on those benchmarks where DirectShare experiences a large accuracy degradation.

Table 2 reports how the performance improves with only 0.5 training epoch for Llama2-7B model. Specifically, in the reading comprehension and knowledge-related tasks mentioned above, PostShare achieves 87.53% of the overall accuracy attained by the original model. Most of the gap between models after DirectShare and the original counterparts can be narrowed via PostShare, especially in BoolQ and RACE datasets.

Last, it is important to emphasize that here we perform post-training with limited training corpus and thus it runs the risk of overfitting when training only for one epoch. For example, PostShare achieves the higher accuracy in BoolQ at 0.3 epoch than at 0.5 epoch (68.29 vs. 66.57). In contrast, as the training epoch increases from 0.5 to 0.9, the accuracy in WinoGrande rises (52.98 vs. 54.39). It means that due to the domain-constrained corpus, overfitting to one specific dataset will potentially compromise the capabilities in other tasks. The in-depth analysis is provided in Appendix F.2.

5.3 Additional Analysis

Statistics of Memory Reduction. Table 3 presents the statistics of the parameter count and memory requirements when applying DirectShare. When sharing 30% parameter sharing in the MHA block, our method achieves 10-13% memory.

Moreover, we find our weight sharing strategy (Section 4.1) also applies to FFN block. We directly observe the weight matrix similarity in FFN and find the concatenation matrix of gate_proj, up_proj and down_proj can be used as matching function for FFN block. Since FFN does not have explainable fine-grained sub-blocks (like attention heads in MHA), we use Traversal Searching method to choose the optimal size of sub-block and find sharing the whole FNN layer works best in the performance maintenance. Finally, when we share 30% of parameter sharing in both MHA and FFN block, the model can save 26-28% GPU memory.

Sharing Ratio	#Params	GPU Memory
	Llama2-7B	
0%	6.74B/100%	17826M/100%
30% MHA	6.09B/90.36%	15512M/87.02%
30% MHA+FFN	4.74B/70.33%	12932M/72.55%
	Llama2-13B	
0%	13.02B/100%	30800M/100%
30% MHA	11.76B/90.32%	27898M/90.58%
30% MHA+FFN	9.21B/70.74%	23002M/74.68%

Table 3: The actual memory savings brought by DirectShare on Llama2 models (recorded during inference on the BoolQ Dataset in OpenCompassv1.0 platform).

Ablation on Head-wise Matching Functions. For weight sharing, the choice of shared heads is critical. In Figure 3, we plot the performance curve on PIQA (Bisk et al., 2020) and OpenBookQA using different head-wise match functions for Baichuan2-7B model. And the corresponding detailed results

Method Ratio=30%	**CMNLI**	**OCNLI**	**AX-b**	**AX-g**	**RTE**	**Wino-Grande**	**BoolQ**	**C-Eval**	**MMLU**	**RACE-middle**	**RACE-high**	**OBQA**	**OBQA-fact**	**CSL**
DirectShare	33.33	32.50	57.07	51.69	49.10	50.18	54.43	26.24	26.53	21.45	21.53	26.00	27.60	51.25
DirectShare + 4bit GPTQ	34.61 ↑ 1.28	30.63 ↓ 1.87	57.79 ↑ 0.72	47.47 ↓ 4.22	49.82 ↑ 0.72	49.12 ↓ 1.06	51.95 ↓ 2.48	21.88 ↓ 4.34	25.38 ↓ 1.15	21.24 ↓ 0.21	21.33 ↓ 0.20	23.40 ↓ 2.60	26.60 ↓ 1.00	50.00 ↓ 1.25

Table 4: Performance of combining weight sharing and quantization on Llama2-7B model.

are presented in Appendix F.3. Notably, using the cosine similarity between the concatenation matrix of W^q and W^k attains the most favorable outcomes. This may be because it guarantees the maximum similarities between attention maps from the model before and after weight sharing. Also, this choice is much more stable and robust in some tasks like reading comprehension(e.g., OpenBookQA).

Robustness on the Model Size. In previous experiments, we adopt our approach in LLMs. Since small-scale models are not highly over-parameterized as large-scale models (Gao et al., 2023), we further verify the effectiveness of our method on smaller models like BERT-base, GPT2-small. For analysis, we set the sharing ratio from 0% to 50% with a step of 10% for the fine-tuned GPT-small model on WMT-14 En-Fr dataset. As shown in Table 5, at a 50% sharing ratio, the GPT-small can still yield a BLEU score of 39.44 without any post-training. Such kind of variance in performance is acceptable that to some degree proves our method is also suitable for small-scale models.

Sharing Ratio	0%	10%	20%	30%	40%	50%
BLEU	**43.62**	42.49	41.95	41.34	39.96	39.44
Meteor	**42.33**	40.75	40.18	38.43	37.21	36.62

Table 5: Robustness on the model size via PostShare (performed on GPT2-small using WMT-14 En-Fr).

Combine Weight Sharing with Quantization. In terms of saving memory, post-quantization employs the strategy of reducing precision in the LLM parameters, while weight sharing aims to reduce the number of parameters. From these two different directions, we suppose integrating weight sharing and quantization may help towards even more memory reduction of LLMs. Hence, we choose GPTQ (Frantar et al., 2022) as a representative and test the effectiveness of applying two techniques in tandem. Specifically, we quantize Llama2-7B model after 30% DirectShare to 4 bit precision. As reported in Table 4, they can be effectively combined with no more than 5 points performance drop.

Combine PostShare with DirectShare. Another interesting research finding is the combination of our DirectShare and PostShare, where PostShare

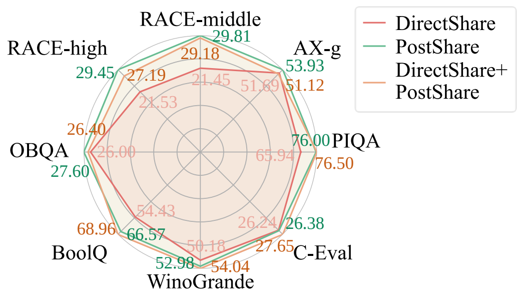


Figure 5: Evaluation results when combining DirectShare with PostShare based on Llama2-7B model.

can play a role in fast performance recovery for DirectShare. Specifically, if we set the sharing ratio to 30% and post-train only 0.5 epoch, the combination based on Llama2-7B performs on par with the PostShare, as Figure 5 shows. It can also be seen that DirectShare+PostShare outperforms in some specific datasets like BoolQ and WinoGrande, which we speculate is due to the mitigation of overfitting problem in PostShare to some extent.

Visualization Study on the Shared Weights. To provide a more detailed explanation of our rationale behind head-wise weight sharing, we conduct a visualization study on the ratios of weight sharing across the MHA layers in two models of different scales (see Appendix F.6). Results indicate the shareable weights distribution across attention heads is similar regardless of the sharing ratio. We also observe a relative balanced sharing ratio across MHA layers than layer-wise weight sharing, which may seem counter-intuitive. However, we find such fine-grained operation on weights has already been used in model pruning (Sun et al., 2023; Ma et al., 2023), constantly superior to layer-wise pruning.

6 Conclusion

In this paper, we illustrate the feasibility of fine-grained weight sharing strategy applied in LLMs, namely, head-wise shareable attention. Consequently, we propose two methods for head-wise weight sharing called DirectShare and PostShare, which are complementary in terms of time and performance. Our DirectShare concentrates on a simple, no-training yet effective sharing strategy, performing competitively with one of the state-of-the-art model pruning methods. PostShare, on the other hand, shows an impressive performance on keeping LLM's capabilities, needing to compromise on time efficiency. Last, we hope our work inspires researchers to explore better fine-grained weight sharing techniques for LLMs.

Limitations

This paper primarily focuses on the head-wise weight sharing in Multi-Head Attention (MHA) block, inspired by the attention map similarity across heads. Although we have explored the feasibility of our proposal weight sharing strategy in the Feed-Forward Network (FFN) block, we only complete downstream evaluation on Baichuan2-7B model. To further verify the effectiveness of applying weight sharing to both MHA and FFN block, we should offer comprehensive experimental validation across different models and compare the results with baselines. We leave it as future work.

Furthermore, the computing resources limited our ability to conduct experiments on LLMs with a model size of more than 13B. Although we hypothesize that our approach can still work in larger models, which proves to have redundant parameters (Frantar and Alistarh, 2023), it is crucial to validate this hypothesis with further exploration.

References

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

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4334–4348.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *ArXiv preprint*, abs/2309.10305.

Srinadh Bhojanapalli, Ayan Chakrabarti, Andreas Veit, Michal Lukasik, Himanshu Jain, Frederick Liu, Yin-Wen Chang, and Sanjiv Kumar. 2021. Leveraging redundancy in attention with reuse transformers. *ArXiv preprint*, abs/2110.06821.

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.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv preprint*, abs/2303.12712.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28.

Xiaodong Chen, Yuxuan Hu, and Jing Zhang. 2024. Compressing large language models by streamlining the unimportant layer. *ArXiv preprint*, abs/2403.19135.

Po-Han Chi, Pei-Hung Chung, Tsung-Han Wu, Chun-Cheng Hsieh, Yen-Hao Chen, Shang-Wen Li, and Hung-yi Lee. 2021. Audio albert: A lite bert for self-supervised learning of audio representation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 344–350. IEEE.

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*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Raj Dabre and Atsushi Fujita. 2019. Recurrent stacking of layers for compact neural machine translation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6292–6299.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations*.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *ArXiv preprint*, abs/2306.03078.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Wikimedia Foundation. Wikimedia downloads.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv preprint*, abs/2210.17323.

Ze-Feng Gao, Kun Zhou, Peiyu Liu, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Small pre-trained language models can be fine-tuned as large models via over-parameterization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 3819–3834.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations*.

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

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence Moss. 2020. OCNLI: Original Chinese Natural Language Inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *ArXiv preprint*, abs/2305.08322.

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. 2022. CSL: A large-scale Chinese scientific literature dataset. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3917–3923.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR.

Peiyu Liu, Ze-Feng Gao, Yushuo Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Enhancing scalability of pre-trained language models via efficient parameter sharing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. 2023. Lightformer: Light-weight transformer using svd-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *ArXiv preprint*, abs/2305.11627.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *ArXiv preprint*, abs/2403.03853.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Kyuhong Shim, Jungwook Choi, and Wonyong Sung. 2023. Exploring attention map reuse for efficient transformer neural networks. *ArXiv preprint*, abs/2301.12444.

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

Sho Takase and Shun Kiyono. 2021a. Lessons on parameter sharing across layers in transformers. *ArXiv preprint*, abs/2104.06022.

Sho Takase and Shun Kiyono. 2021b. Rethinking perturbations in encoder-decoders for fast training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5767–5780, Online. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Shicheng Tan, Weng Lam Tam, Yuanchun Wang, Wenwen Gong, Yang Yang, Hongyin Tang, Keqing He, Jiahao Liu, Jingang Wang, Shu Zhao, et al. 2023. Gkd: A general knowledge distillation framework for large-scale pre-trained language model. *ArXiv preprint*, abs/2306.06629.

Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4821–4836.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *ArXiv preprint*, abs/2206.07682.

Siyue Wu, Hongzhan Chen, Xiaojun Quan, Qifan Wang, and Rui Wang. 2023. Ad-kd: Attribution-driven knowledge distillation for language model compression. *ArXiv preprint*, abs/2305.10010.

Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5466–5473.

Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. 2019. Sharing attention weights for fast transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 5292–5298.

Canwen Xu and Julian McAuley. 2023. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10566–10575.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *ArXiv preprint*, abs/2402.11187.

Ziqing Yang, Yiming Cui, Xin Yao, and Shijin Wang. 2022. Gradient-based intra-attention pruning on pre-trained language models. *ArXiv preprint*, abs/2212.07634.

Chengxuan Ying, Guolin Ke, Di He, and Tie-Yan Liu. 2021. Lazyformer: Self attention with lazy update. *ArXiv preprint*, abs/2102.12702.

Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. 2022. Minivit: Compressing vision transformers with weight multiplexing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12145–12154.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *ArXiv preprint*, abs/2303.18223.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. 2021. Deepvit: Towards deeper vision transformer. *ArXiv preprint*, abs/2103.11886.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *ArXiv preprint*, abs/1710.01878.

A Detailed Explanation for DirectShare

Algorithm 1 summarizes the procedure of DirectShare:

(1) Matching (line 1-13)

We first initialize the buffer $\mathcal{D}_\tau$ **(line 1)** which is then used to store each candidate shareable attention head pairs. Next, the iterative process of matching begins:

Prepare Candidate Pairs (line 2-9): Given the number of layers $\mathcal{L}$ and the number of heads per MHA block $\mathcal{H}$, we can construct candidate attention head pairs for sharing. For each pair, we then record the layer index and the head index accordingly, i.e., $index_i$ and $index_m$ respectively.

Calculate Similarity (line 10-13): We use 10 randomly selected samples from Wikipedia as the calibration samples for calculating the weight matrix similarity. We present the design of head-wise match function in Section 4.1. The final matching score $\mathcal{S}_{cos}$ is obtained by averaging across samples **(line 10)**. For every attention head, our approach selects the most matching counterpart according to $\mathcal{S}_{cos}$ **(line 11-12)** and stores them as pairs in the buffer $\mathcal{D}_\tau$ **(line 13)**.

(2) Weight Sharing (line 14-16)

We sort the candidate pairs in $\mathcal{D}_\tau$ in descending order of their matching scores s_m **(line 14)**. Then, the top-N pairs $\mathcal{N}$ are selected according to the desired sharing ratio α **(line 15)**. Finally, for each selected head pair, we keep the weight values of the head in former layer unchanged and select to change the weight values of the head in latter layer **(line 16)**, resulting in the final model $\mathcal{M}^*$.

B Relation between Matrix Weights and Attention Map Similarities

Our visual analysis (in Section3.2) can intuitively illustrate that the shared head pair with the highest similar weight matrices to a large degree exhibits highly similar attention maps across different datasets. Aside from this, we also provide a metric to measure the relation:

$$Degree = \frac{\#num\ of\ matched\ heads}{\#num\ of\ samples} \tag{6}$$

Algorithm 2: Weight_Share Function

```
Input: top-N matched head pairs $\mathcal{N}$,
       Original LLM $\mathcal{M}$
Output: The LLM $\mathcal{M}^*$ after weight sharing
1  Initialize one dict $\mathcal{D}$;
2  for $pair$ in $\mathcal{N}$ do
3      $index_i, index_j \leftarrow pair$;
4      $(layer_i, i) \leftarrow index_i$;
5      $(layer_j, j) \leftarrow index_j$;
6      if $layer_i < layer_j$ then
7          $\mathcal{D}[index_i] = index_j$;
8      else
9          $\mathcal{D}[index_j] = index_i$;
10 Sort $\mathcal{D}$ by the key;
11 $\mathcal{M}^* \leftarrow \mathcal{M}$;
12 for $head_i, head_j$ in $\mathcal{D}$ do
13     $\mathcal{M}^* \leftarrow$ tie_weight($\mathcal{M}^*$, $head_i$, $head_j$);
14 return $\mathcal{M}^*$;
```

Each time we use 100 randomly selected samples from each dataset to calculate the average cosine similarity value and repeat 5 times. According to the sharing ratio α, we will respectively select the top-K shared head pairs based on the weight matrix similarity and attention map similarity (calculated cosine similarity as metric), denoted as Set A and B. The number of matched heads is calculated as the cardinality of the intersection of Set A and Set B, denoted as $|A \cap B|$. From Table 6, it can be seen the matched degree is relatively high, so it is reasonable for us to do weight sharing directly across heads.

Dataset	WMT-14	CQA	WSC	**Average**
Llama2-7B	100%	91.30%	100%	97.10%

Table 6: The degree metric measured to quantify the relation between matrix weights and attention map similarities. CQA stands for CommonsenseQA benchmark.

C Implementation Details

In this section, we will provide additional information about our experimental implementation.

C.1 Baselines

To our knowledge, there is no existing baseline for our methods, due to the absence of prior research on fine-grained weight sharing for LLMs. To provide a comprehensive demonstration of the effectiveness of our DirectShare, we can choose another important memory-efficient method of a different category for comparison. Here, we select two model pruning methods applied in LLMs: one classical model pruning method Magnitude Pruning and one state-of-the-art structured pruning method LLM-Pruner. We do not consider unstructured pruning methods in this paper since they can not achieve real memory reduction without specialized hardware or software. Appendix G lists more comparison results with some baselines modified to support LLMs.

Based on the results presented in Table 1, 7, 8, 10, it is evident that our DirectShare performs on par with one of the prior best structured pruning methods regarding the overall performance, superior to the standard magnitude pruning. Consequently, we claim that designing such a fine-grained (i.e., head-wise) weight sharing strategy with a specific focus on LLMs is indeed simple but effective and this would be a good direction for future work.

C.2 Benchmarks

Logical and Common Sense Reasoning. In the domain of reasoning, we consider two Chinese natural language inference benchmarks and three English benchmarks: CMNLI (Xu et al., 2020), OCNLI (Hu et al., 2020), along with AX-b, AX-g and RTE from SuperGLUE (Wang et al., 2019).
Natural Language Understanding (NLU). In this field, we cover multiple tasks, including RACE (Lai et al., 2017) and OpenBookQA (Mihaylov et al., 2018) for reading comprehension, CSL (Li et al., 2022) for content summary and TNEWS (Xu et al., 2020) for content analysis.
Knowledge-related Tasks. We perform evaluations regarding knowledge on various datasets: WinoGrande (Levesque et al., 2012) about language, BoolQ (Clark et al., 2019) testing knowledge question answering, C-Eval (Huang et al., 2023) and MMLU (Hendrycks et al., 2021) standing for two comprehensive examination benchmarks.

C.3 Post-training Details

For carrying out the post-training process, we employ the code framework from LLaMA-Factory repository[2] with DeepSpeed ZeRO-1[3]. The Adam optimizer with a learning rate of 5e-5 is used in our experiment and the parameter values assigned during training are $\beta_1 = 0.9$ and $\beta_2 = 0.95$. For Llama 2-7B model, we set the batch size to 32. While for Llama 2-13B model, the batch size of training is only 8 subject to the limited computational resources. Besides, the maximum context size and γ are set to 4096 and 0.5, respectively.

[2]https://github.com/hiyouga/LLaMA-Factory

[3]Because of our designed special loss function in the post-training stage, only DeepSpeed ZeRO-1 can work.

Ratio	Method	CMNLI	OCNLI	AX-b	AX-g	RTE
0%	**Baichuan2-7B**	33.37	41.88	51.90	50.28	57.40
10%	Magnitude	33.11	33.12	**55.62**	50.84	55.96
	LLM-Pruner	**37.31**	40.62	49.18	50.00	60.65
	DirectShare	33.00	**41.25**	49.55	**51.12**	60.29
30%	Magnitude	32.97	31.25	48.28	**51.97**	46.57
	LLM-Pruner	**34.20**	**34.38**	47.55	50.84	**51.26**
	DirectShare	32.97	30.63	**54.71**	51.69	49.82
0%	**Baichuan2-13B**	33.21	40.62	59.69	50.59	44.77
10%	Magnitude	33.21	31.25	55.62	48.60	46.93
	LLM-Pruner	**33.66**	36.88	**58.51**	49.72	47.65
	DirectShare	33.23	**40.00**	53.71	**53.37**	**53.07**
30%	Magnitude	**33.21**	30.00	50.91	48.03	43.32
	LLM-Pruner	33.04	**36.88**	**55.71**	**50.28**	44.04
	DirectShare	33.11	30.00	54.98	50.00	**45.13**

Table 7: Evaluation results on reasoning tasks when applying DirectShare to Baichuan2 models.

Ratio	Method	RACE-middle	RACE-high	OBQA	CSL	TNEWS
0%	**Baichuan2-7B**	51.04	52.63	32.20	66.25	28.60
10%	Magnitude	24.37	28.13	30.20	57.50	**27.60**
	LLM-Pruner	25.42	35.36	**32.60**	61.25	26.05
	DirectShare	**50.49**	**48.46**	28.20	**63.75**	26.23
30%	Magnitude	21.80	21.67	**27.60**	**57.50**	13.66
	LLM-Pruner	22.56	22.67	27.40	53.12	**21.31**
	DirectShare	**25.14**	**23.44**	**27.60**	52.50	18.40
0%	**Baichuan2-13B**	68.94	67.27	42.20	63.12	28.96
10%	Magnitude	25.56	26.33	26.20	45.62	11.38
	LLM-Pruner	41.71	46.80	**32.40**	62.50	**29.23**
	DirectShare	**47.56**	**49.34**	31.20	**64.38**	22.22
30%	Magnitude	**24.58**	**24.58**	25.40	50.62	6.65
	LLM-Pruner	22.63	21.81	**26.80**	**55.00**	**24.13**
	DirectShare	22.14	23.99	26.60	53.13	17.58

Table 8: NLU abilities of Baichuan2 models after DirectShare.

D Experimental Results based on Baichuan 2 Models

We re-implement Magnitude Pruning and LLM-Pruner with their public code to accommodate Baichuan2 models.

D.1 Logical and Common Sense Reasoning

Table 7 presents a comparison on five datasets about reasoning abilities for three memory-efficient methods performed on the Baichuan2 models. Our results show that compared to NLU and knowledge-related abilities (listed in Table 8,10), DirectShare can indeed maintain its reasoning abilities to a large

Ratio	Method	WinoGrande	BoolQ	C-Eval	MMLU	RACE-middle	RACE-high	OBQA	OBQA-fact
0%	**Llama 2-13B**	55.44	71.50	40.17	55.81	60.24	58.03	42.40	60.00
30%	DirectShare	50.18	59.36	22.30	30.79	26.53	27.53	27.40	27.80
	PostShare*	53.68 ↑ 3.50	71.25 ↑ 11.89	25.80 ↑ 3.50	33.90 ↑ 3.11	32.03 ↑ 3.30	29.07 ↑ 1.54	33.60 ↑ 6.20	38.80 ↑ 11.00

Table 9: Performance of PostShare based on the Llama2-13B backbone. * means choosing relatively good performance across different training steps.

Ratio	Method	WinoGrande	BoolQ	C-Eval	MMLU
0%	**Baichuan2-7B**	54.04	63.30	56.19	54.65
10%	Magnitude	50.18	57.06	34.70	45.47
	LLM-Pruner	50.53	**59.30**	48.14	**51.78**
	DirectShare	**51.58**	58.01	**50.41**	49.96
30%	Magnitude	49.12	**55.41**	**23.91**	24.36
	LLM-Pruner	51.23	48.93	22.11	**25.62**
	DirectShare	**51.58**	51.53	21.86	24.05
0%	**Baichuan2-13B**	56.14	67.00	59.21	59.58
10%	Magnitude	50.53	40.55	25.22	25.55
	LLM-Pruner	51.23	**65.87**	49.60	51.49
	DirectShare	**53.33**	61.04	**53.65**	**52.60**
30%	Magnitude	50.18	50.09	**25.35**	24.66
	LLM-Pruner	**50.53**	**59.42**	21.09	**24.95**
	DirectShare	48.77	40.83	23.25	24.82

Table 10: Results on knowledge-related tasks of Baichuan2 models after DirectShare.

extent. Specifically, at 30% ratio, DirectShare remains competitive with LLM-Pruner.

D.2 Natural Language Understanding

Table 8 presents the performance for each NLU task discussed in Section 5.2.1 when applying DirectShare to Baichuan2 models. Consistent with the experiments on Llama2-7B and Llama2-13B models, similar performance drop exists. Thus, at the cost of post-training time, our PostShare can narrow the gap observed across the majority of datasets. With regard to individual datasets, it remains to be seen if the gap can be largely recovered given the best training epoch[4].

D.3 Knowledge-related Tasks

The results of Baichuan2 models on knowledge-related tasks are shown in Table 10. Similar decline appears in Llama2-7B/13B models as well.

E PostShare on Llama 2-13B Model

In addition to Llama2-7B, we also experiment with Llama2-13B to evaluate PostShare (See Table 9). Compared to Llama2-7B, the best training epoch

[4]We speculate that it may be attributed to overfitting issue. Furthermore, as the model size increases, it becomes increasingly difficult to determine the optimal training epoch for effectively mitigating overfitting.

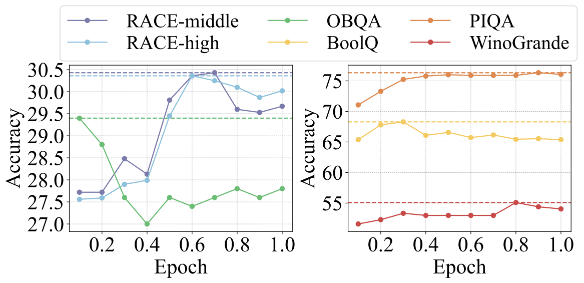


Figure 6: Accuracy across different training steps during PostShare.

Epoch	RACE-middle	RACE-high	OBQA	BoolQ	PIQA	Wino-Grande
0.10	27.72	27.56	**29.40**	65.38	71.06	51.58
0.20	27.72	27.59	28.80	67.80	73.29	52.28
0.30	28.48	27.90	27.60	**68.29**	75.24	53.33
0.40	28.13	27.99	27.00	66.09	75.79	52.98
0.50	29.81	29.45	27.60	66.57	76.00	52.98
0.60	30.36	**30.36**	27.40	65.72	75.90	52.98
0.70	**30.43**	30.25	27.60	66.15	75.90	52.98
0.80	29.60	30.10	27.80	65.44	75.90	**55.09**
0.90	29.53	29.87	27.60	65.54	**76.33**	54.39
1.00	29.67	30.02	27.80	65.38	76.06	54.04

Table 11: Accuracy across different training steps during PostShare.

on Llama2-13B is much smaller: approximately hundreds of training steps is enough, otherwise it may suffer from overfitting issue. However, the overfitting problem seems to be obvious as model size increases, resulting in the challenge with regard to choosing the best training epoch.

F More Analysis

F.1 Training Time Costs of PostShare

For Llama2-7B model, our full-parameter post-training in English Wikipedia corpus uses around 11.27 hours for 0.5 epoch due to our limited computational resources. We list the post-training experimental settings in Appendix C.3. In the same settings, the original full-parameter post-training requires 10.78 hours for 0.5 epoch, which indicates our PostShare does not increase much training costs (gap≈30min).

Sharing Ratio	**5%**		**10%**		**15%**		**20%**		**25%**		**30%**		**35%**	**40%**
Dataset	PIQA	OBQA	PIQA	OBQA	PIQA	OBQA	PIQA	OBQA	PIQA	OBQA	PIQA	OBQA	OBQA	OBQA
W^q	74.92	29.2	74.97	27.5	73.29	27.8	70.89	27.7	64.64	27.5	58.43	25.5	24.4	25.7
W^k	74.92	28.7	74.27	27.6	71.71	27.7	70.35	27.6	68.77	**27.6**	64.36	27.2	27.6	26.9
W^v	74.92	28.1	74.48	27.7	73.29	26.7	70.46	**28.5**	68.39	25.6	60.17	23.1	23.9	22.5
W^q, W^k, W^v	71.71	27.6	63.55	27.8	54.03	26.8	50.16	24.5	51.41	25.5	51.09	25.5	**29.0**	25.3
$W^q\|\|W^k\|\|W^v$	74.59	**34.7**	74.59	**30.0**	73.45	**30.3**	70.73	28.2	66.59	**27.6**	63.33	27.6	27.1	25.0
$W^q\|\|W^k$(Ours)	**75.84**	33.9	**75.30**	28.2	**74.54**	27.5	**73.01**	27.3	**69.37**	27.5	**65.56**	**28.0**	27.6	**28.6**

Table 12: Results on PIQA and OBQA with different head-wise matching functions for Baichuan2-7B model.

	CMNLI	OCNLI	AX-b	AX-g	RTE	RACE-middle	RACE-high	OBQA	CSL	TNEWS	Wino-Grande	BoolQ	C-Eval	MMLU	Avg.
Choosing A	33.33	32.50	57.07	51.68	49.10	21.45	21.53	26.00	51.25	20.22	50.18	54.43	26.24	26.53	**37.25**
Choosing B	33.59	31.25	51.09	53.09	53.43	25.00	25.53	22.20	49.38	12.20	52.63	57.22	28.27	25.63	37.18

Table 13: Performance comparison between choosing different shared weights for a pair of matched heads A, B.

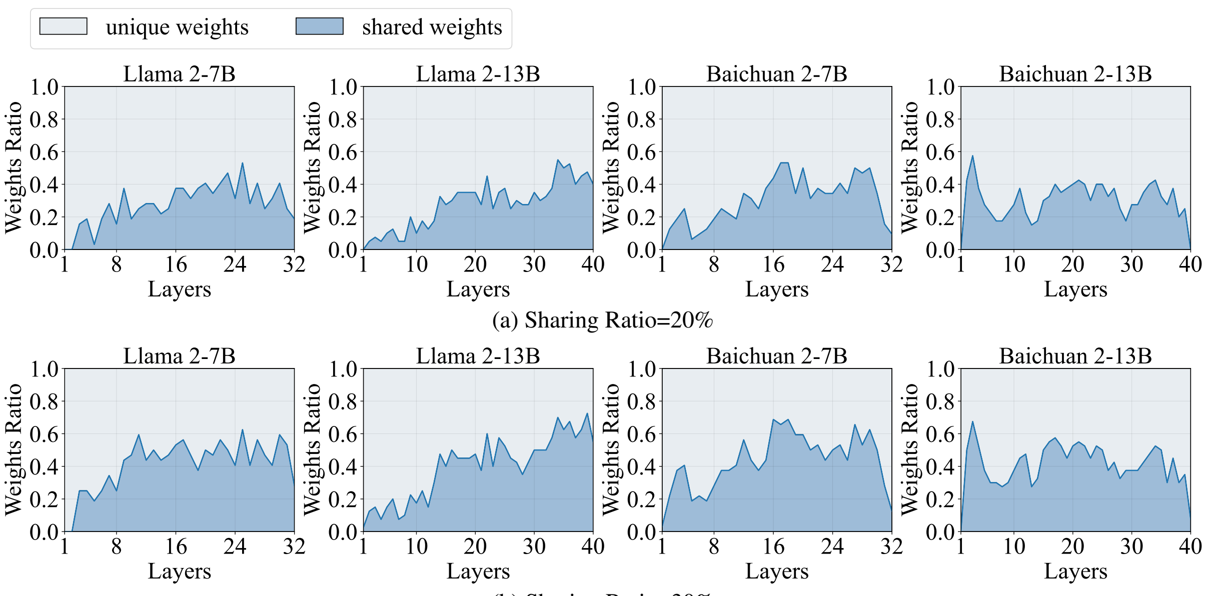


Figure 7: Ratios of weight sharing across the MHA Layers in Llama2-7B/13B and Baichuan2-7B/13B.

F.2 Overfitting Phenomenon in PostShare

Figure 6 shows the performance curves on different kinds of datasets across various post-training steps. Remarkably, our PostShare requires no more than 1 epoch that can push the selected weights closer for sharing while keeping the performance. However, we observe the slight overfitting phenomenon in PostShare, i.e., the capabilities initially improve and then experience a slight decline. Besides, it is clear that the turning point about performance varies with datasets. Detailed statistical data are provided in Table 11.

F.3 Impact of Different Head-wise Matching Functions

The selection of shared heads plays a crucial role in weight sharing. An ablation experiment for this is shown in Table 12.

F.4 How to Choose the Shared Weights?

We select the weight value from the head in the former layer as the shared weights. The reason behind is that we observe the weight values of the earlier layers are more sensitive in our experiments, which is also supported by previous studies (Men et al., 2024; Chen et al., 2024). Thus, we tend to keep the weight values of the head in former layer unchanged. We also additionally make a performance comparison between choosing different shared weights for a pair of matched heads A, B. Suppose head A is in the former layer and head B is in the latter layer. In Table 13, it can be seen that the performance gap is small on the downstream tasks whether choosing A or B, but the overall performance is a little better when choosing the head weight in the former layers as the shared weights.

Method		CMNLI	OCNLI	AX-b	AX-g	RTE	RACE-middle	RACE-high	OBQA	CSL	TNEWS	Wino-Grande	BoolQ	C-Eval	MMLU
DirectShare	Ratio=10%	33.00	32.50	54.17	51.97	50.90	28.34	28.96	28.20	54.37	20.86	52.63	67.74	28.75	43.43
GQA	group size=2	32.20	27.50	58.33	50.00	47.29	25.21	25.61	22.20	46.25	6.92	50.53	37.83	25.68	25.80
DirectShare	Ratio=30%	33.33	32.50	57.07	51.68	49.10	21.45	21.53	26.00	51.25	20.22	50.18	54.43	26.24	26.53

Table 14: Comparison results with GQA across various datasets.

	Ratio	CMNLI	OCNLI	AX-b	AX-g	RTE	RACE-middle	RACE-high	OBQA	CSL	TNEWS	Wino-Grande	BoolQ	C-Eval	MMLU
Llama2-7B	0%	32.98	33.12	53.53	55.34	49.82	33.15	35.51	31.80	55.62	20.22	54.04	70.67	32.20	46.69
DirectShare	10%	33.00	32.50	54.17	51.97	50.90	28.34	28.96	28.20	54.37	20.86	52.63	67.74	28.75	43.43
SEQ	10%	31.55	37.50	49.82	51.40	52.35	28.62	28.39	27.00	57.50	15.39	51.23	64.77	30.40	39.69
DirectShare	30%	33.33	32.50	57.07	51.68	49.10	21.45	21.53	26.00	51.25	20.22	50.18	54.43	26.24	26.53
SEQ	30%	31.72	32.50	47.55	50.00	46.93	22.63	22.30	26.20	50.62	9.83	49.47	52.60	22.05	23.49

Table 15: Comparison results with SEQ across various datasets.

F.5 Latency Evaluations

Our paper aims to reduce the memory requirements for LLMs instead of accelerating. Since weight sharing does not change the total matrix multiplications, DirectShare can theoretically maintain the original model inference speed but our current implementation of head-wise weight sharing is hard to save memory and speed up at the same time, which may be a programming skill issue. Table 16 reports the inference speed tested under the test set of WikiText2. We will consider further optimization in future work to achieve a balance between memory and speedup.

Model	Sharing Ratio	Speed
Llama2-7B	0%	41.70 tokens/s
	30% MHA	8.52 tokens/s
	30% MHA+FFN	8.34 tokens/s
Llama2-13B	0%	24.20 tokens/s
	30% MHA	5.63 tokens/s
	30% MHA+FFN	5.67 tokens/s

Table 16: The inference speed of our method tested on a single A800-80GB GPU.

F.6 Visualization Study on the Shared Weights

As depicted in Figure 7, the distribution of ratios of shared weights across attention heads is similar regardless of the sharing ratio.

G More Comparison Results

G.1 Compared with Grouped Query Attention

Grouped Query Attention (GQA) has been widely adopted in LLMs to reduce the number of attention computations required, by grouping the query vectors into a smaller number of groups. Although GQA might be less flexible in choosing the reduction ratio of parameters, we also provide a performance comparison with our method. To equip the original Llama2-7B model with GQA, we perform the standard conversion proposed by Ainslie et al. (2023) and uses neighbor grouping with group size=2 to merge heads. The performance comparison is shown in Table 14. Furthermore, we compute the average scores per category (Reasoning, NLU, Knowledge) along with the average scores across all benchmarks (Avg.) to provide a more intuitive comparison in Table 17. Overall, our method demonstrates superior performance over GQA.

		Reasoning	NLU	Knowledge	Avg.
DirectShare	Ratio=10%	44.51	32.15	48.14	41.60
GQA	group size=2	37.06	25.24	34.96	32.42
DirectShare	Ratio=30%	44.74	28.09	39.35	37.39

Table 17: Average comparison results with GQA.

G.2 Compared with Modified Layer-wise Weight Sharing

We state the reason why choosing model pruning works as baselines in Section C.1. Prior layer-wise weight sharing techniques are proposed to apply to small-scale models like BERT. Considering one representative layer-wise weight sharing method (Takase and Kiyono, 2021b), we need to modify it to support LLMs, denoted as SEQ. For fair comparison, we perform layer-wise weight sharing in latter 60% layers instead of all layers. The overall performance comparison based on Llama2-7B model is presented in Table 15. Moreover, we calculate the overall average scores across all benchmarks (Avg.), the average scores for each category (Reasoning, NLU, Knowledge), and the average performance percentages relative to the original model across all benchmarks (Per.). The results in Table 18 indicate that our head-wise weight sharing method performs better than layer-wise weight sharing.

	Ratio	Avg.	Per.	Reasoning	NLU	Knowledge
Llama2-7B	0%	43.71	100%	44.96	35.26	50.90
DirectShare	10%	41.60	95.17%	44.51	32.15	48.14
SEQ	10%	40.74	93.21%	44.52	31.18	46.53
DirectShare	30%	37.39	85.54%	44.74	28.09	39.35
SEQ	30%	34.97	80.00%	41.68	26.32	36.90

Table 18: Average comparison results with SEQ.

H Generations from Models after Weight Sharing

Table 19, 20 and 21 show more examples of the models after head-wise weight sharing. We present the generation results of both the shared model via DirectShare and PostShare.

Model	Generated Responses
Llama2-7B w/ DirectShare	The future is going to be one that presents many challenges, but also a great deal of opportunity. Everybody knows this, and it's why we have been working so hard on the newest version of the future. The problem with the future is that it's not always easy to see what you are doing wrong. It can be difficult to find
	Neural Sequential Model, especially transformers, are a powerful tool for the analysis of time-series data. surely, they can be used to predict the future. Their use is not limited to the financial industry. In this article, we will discuss the benefits and limitations of the
Llama2-7B w/ PostShare	The future is going to be one that presents many challenges, and the world will need a new generation of leaders. The United Nations has identified the following as key issues for the future: Climate change, Food security, Energy security, Water scarcity, Global health, Human rights, International law, Peacekeeping, Sustainable
	Neural Sequential Model, especially transformers, is a type of artificial neural network that uses a sequence of vectors to represent the input and output. The model was introduced by Geoffrey Hinton in 1987. It has been used for speech recognition, handwriting recognition,

Table 19: Responses from Llama2-7B (Sharing Ratio=30%).

Model	Generated Responses
Baichuan2-7B w/ DirectShare (English)	There are many reasons to support why perseverance is of great importance. The first and foremost, it's the only way you can get better at what ever your doing! It also gives us an opportunity to learn from our mistakes or
	Her eyes shone with happiness. "I'm so happy to see you," She said, "and I am very proud of the work that we have done together in this project and it is a great honor for me as an artist because there are many people who has been working on these projects over the years but not all of them were able to do what was needed from us at any given time when things got out of hand or if they had lost their way"
Baichuan2-7B w/ DirectShare (Chinese)	如何健康的生活 1. 注意休息,保证睡眠。2.合理饮食:多吃水果、蔬菜等。3.适当运动(如散步) 4.保持乐观心态5.多饮水6.避免劳累7.不要熬夜8.劳逸结合9.注意安全10.尊重生命
	想要成功需要把握什么?1. 首先,要有一个好的心态。2.其次,要有一定的经济基础3.接着就是自己的实力了4. 最后一个条件是:自己一定要努力!

Table 20: Responses from Baichuan2-7B (Sharing Ratio=30%).

Model	Generated Responses	Reference
GPT2-small w/ PostShare	In the letter, Mr Snowden said he believed the support of the international community could persuade the US government to abandon criminal charges against him.<to-fr> Dans la lettre, M. le député a dit que l'appui de la communauté internationale pourrait convaincre le gouvernement américain de abandonner la charge pénale contre lui.	Dans sa lettre, M. Snowden a écrit qu'il pensait que le soutien de la communauté internationale pourrait persuader le gouvernement américain d'abandonner les charges pénales retenues contre lui.
	The letter extends an offer to cooperate with German authorities "when the difficulties of this humanitarian situation have been resolved."<to-fr> La lettre élargit une invitation à coopérer avec les autorités allemands lorsque les difficultés de cette situation humanitaire sont réglées.	La lettre avance une offre de coopération avec les autorités allemandes « lorsque les difficultés de cette situation humanitaire auront été résolues ».
	The first test plane was unveiled in March and took flight for the first time in September after months of delays.<to-fr> Le premier étudiant a été démontré en mars et a fait l'avion pour la première fois après des mois de retard.	Le premier avion d'essai a été dévoilé en mars et s'est envolé pour la première fois en septembre après des mois de retard.

Table 21: Responses from GPT2-small (Sharing Ratio=30%).