Cross-layer Attention Sharing for Pre-trained Large Language Models

Yongyu Mu[1]*, Yuzhang Wu[1], Yuchun Fan[1], Chenglong Wang[1], Hengyu Li[1], Jiali Zeng[2], Qiaozhi He, Murun Yang[1], Fandong Meng[2], Jie Zhou[2], Tong Xiao[1]† and **Jingbo Zhu[1]**

[1]NLP Lab, School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
lixiaoyumu9@gmail.com
{lemonzeng,fandongmeng,withtomzhou}@tencent.com
{xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract

To enhance the efficiency of the attention mechanism within large language models (LLMs), previous works primarily compress the Key-Value cache or group attention heads, while largely overlooking redundancy between layers. Our comprehensive analyses across various LLMs show that highly similar attention patterns persist within most layers. It's intuitive to reduce the redundancy by sharing attention weights across layers. However, further analysis reveals two challenges: (1) Directly sharing the weight matrix without carefully rearranging the attention heads proves to be ineffective; (2) Shallow layers are vulnerable to small deviations in attention weights. Driven by these insights, we introduce LiSA, a lightweight substitute for self-attention in well-trained LLMs. LiSA employs tiny feed-forward networks to align attention heads between adjacent layers and low-rank matrices to approximate differences in layer-wise attention weights. Evaluations encompassing 13 typical benchmarks demonstrate that LiSA maintains high response quality in terms of accuracy and perplexity while reducing redundant attention calculations within $53\% - 84\%$ of the total layers. Our implementations of LiSA achieve a $6\times$ compression of Q and K matrices within the attention mechanism, with maximum throughput improvements 19.5%, 32.3%, and 40.1% for LLaMA3-8B, LLaMA2-7B, and LLaMA2-13B, respectively. Our code is available at https://github.com/takagi97/lisa.

1 Introduction

Many Transformer models are over-parameterized, leading to significant redundancy across various model components, including attention mechanisms (Tay et al., 2023), feed-forward networks (Pires et al., 2023), layers (Matsubara et al., 2023), and others (Lan et al., 2020; Jaegle et al., 2022; Han et al., 2020). When entering the era of large language models (LLMs), the parameters have extremely expanded. For example, comparing open-source pre-trained models between BERT$_{BASE}$ (Devlin et al., 2019) and Bloom-176B (Scao et al., 2022), the number of parameters has grown nearly $1600\times$, let alone the commercial closed-source ones. Consequently, the redundancy of these models also increases at a gallop.

One of the typical instances is that though the self-attention mechanism consumes unbearably massive memory and computation when tackling long sequences in LLMs, its crucial weight matrix is extremely sparse (Liu et al., 2023a; Zhang et al., 2023; Kitaev et al., 2020), which means substantial computational resources predominantly contribute to marginal effects. Thus, recently, reducing the redundancy within the self-attention of LLMs has become a continually appealing focus. One line of work along this research is reducing the Key-Value (KV) cache by cutting down useless tokens (Liu et al., 2023a; Zhang et al., 2023; Xiao et al., 2024) or compressing the representation of KV cache (DeepSeek-AI et al., 2024; Kang et al., 2024). Others attempt to prune the attention heads via clustering (Agarwal et al., 2024) or sparsity predictor (Liu et al., 2023b).

Indeed, most previous works focus on reducing intra-layer redundancy within LLMs' attention mechanisms. However, inter-layer redundancy—specifically whether it's necessary to calculate attention at every layer—has been overlooked. Efforts contributing to this area are non-trivial, as scaling LLMs leads to more stacked layers, which might sharply increase inter-layer

*Work was done when Yongyu Mu was interning at Pattern Recognition Center, WeChat AI, Tencent Inc.

†Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 656–688, 2026. https://doi.org/10.1162/TACL.a.616
Action Editor: Xavier Carreras. Submission batch: 8/2024; Revision batch: 8/2025; Published 5/2026.
© 2026 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

redundancy. In this work, we aim to answer the questions: *To what extent does the redundancy of attention exist across layers in LLMs, and what hinders us from reducing this redundancy?*

We start with a pioneer similarity analysis of each sub-module of the attention mechanism in LLMs. A widespread observation is that the attention weights of most layers are highly similar, especially in adjacent layers of large models. Inspired by the efforts of sharing similar parameters or activation (Ainslie et al., 2023; Xiao et al., 2019; Gomez et al., 2017), a natural next step is to reuse the attention weight matrices calculated by shallow layers and share them with others. Yet, our analysis shows that this naïve approach inherently faces two main challenges:

- Directly sharing the weight matrix without carefully rearranging attention heads is ineffective. Since heads lack positional relationships, directly sharing them is akin to random permutation, adversely impacting similarity. Indeed, most heads can be aligned with a highly similar one in the shared matrix, making it crucial to align them before sharing.
- Shallow layers are sensitive to attention weights. Even small deviations can cause performance collapse. Therefore, a remedy for differences is necessary.

To address these challenges, we take a further step by presenting a simple, lightweight, and **L**earnable **S**haring **A**ttention mechanism (LiSA) for existing well-trained LLMs. LiSA involves two key components. The first is the *attention heads alignment* module, wherein we align the attention heads in the shared matrix with ones of the current layer to reuse the weights from the most similar heads. The second is the *difference compensation* module, which can approximate the differences of attention weight matrices in two layers, thus preventing performance loss caused by tiny deviations. Experimental results on 13 typical benchmarks show that applying LiSA to more than half of the total layers achieves performance comparable to the original model, even on challenging tasks like mathematical reasoning, while requiring only 0.46% to 1.64% of the parameters to be trained. In terms of efficiency, LiSA significantly reduces redundant attention calculations within $53\%-84\%$ of the total layers via compressing both Q and K matrices by $6\times$. Consequently, LiSA achieves throughput improvements of 19.5% for LLaMA3-8B, 32.3% for LLaMA2-7B, and 40.1% for the larger 13B, with the latter two underscoring LiSA's scaling benefits.

2 Background and Related Work

Most methods that enhance the efficiency of Transformer models generally reduce redundancy in parameters, structures, and other aspects. These methods include knowledge distillation (Jiao et al., 2020; Sun et al., 2020; Lin et al., 2021; Sun et al., 2020), pruning (Voita et al., 2019; Fan et al., 2020; Gordon et al., 2020; Mao et al., 2020; Sanh et al., 2020), quantization (Shen et al., 2020; Dettmers et al., 2022; Kim et al., 2021), neural architecture search (Wang et al., 2020a; Xu et al., 2021, 2022), and hardware-aware optimization (Dao et al., 2022; Dao, 2024; Ham et al., 2020; Fang et al., 2022). In this work, we focus on the redundancy within the attention mechanism. We first review the efficient attention methods used in previous Transformer models and then summarize those specifically designed for LLMs.

2.1 Standard Transformer Models

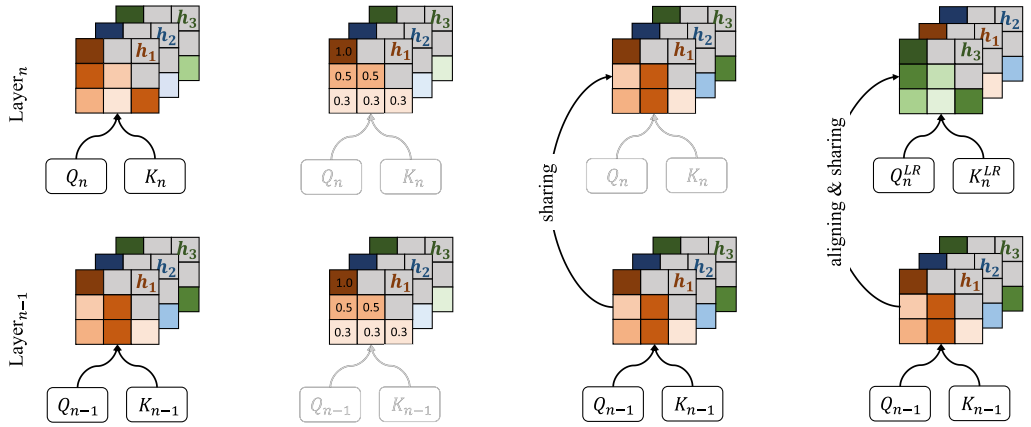
Let $H \in \mathbb{R}^{l\times d}$ represent the hidden state, where l is the sequence length and d is the dimension of the hidden states. The scaled dot-product multi-head attention (MHA), utilizing h attention heads in d_k dimensions, is defined as follows:

$$\text{MHA}(H) = \text{Concat}(P_1 H W_1^V, \ldots, P_h H W_h^V) W^O \quad (1)$$

$$\text{where } P_i = \text{Softmax}\underbrace{\left[\frac{H W_i^Q (H W_i^K)^T}{\sqrt{d_k}}\right]}_{A} \quad (2)$$

where $P_i \in \mathbb{R}^{l\times l}$ is the attention weight matrix, A is the intermediate result before Softmax$(\cdot)$, and three linear projections $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times hd_k}$ process the representations into Q, K, and V matrices. Finally, the output linear projection $W^O \in \mathbb{R}^{hd_k\times d}$ integrates representations from different heads into a single output.

Numerous studies have focused on identifying and reducing redundancy within the components of the attention mechanism, including sparse attention activation P (Luong et al., 2015; Sperber et al., 2018; Parmar et al., 2018; Ainslie et al., 2020; Roy et al., 2021; Kitaev et al., 2020), pruning and grouping attention heads (Michel et al.,



(a) Standard Attention (b) Average Attention (c) Directly Sharing Attention (d) LiSA Attention

Figure 1: Comparison of different attention models. Layer$_n$ stands for a Transformer layer while h_1, h_2, and h_3 represent three attention heads. *Standard attention* individually calculates attention scores at each layer by employing Q_n and K_n matrices. *Average attention* assigns uniform weights across all token positions, thus eliminating Q and K matrices. *Directly sharing attention* reuses the raw weight matrix from the front layer but overlooks varied head weights across different layers. Our method, LiSA attention, not only aligns attention heads but also compensates for layer-wise weight differences leveraging low-rank Q_n^{LR} and K_n^{LR} matrices, thus maximally preserving the original performance while introducing only a few additional training parameters.

2019; Voita et al., 2019), compressing representations Q, K, and V (Liu et al., 2018; Katharopoulos et al., 2020; Wang et al., 2020b). In addition to these intra-layer methods, some works aim to reduce layer-wise redundancy by reusing parameters (Pires et al., 2023) or attention weights P (Xiao et al., 2019), and skipping unnecessary layers (Teerapittayanon et al., 2016).

The most similar work to ours is SAN (Xiao et al., 2019), as it leverages the similarity of attention weights P across multiple layers and directly shares them in neural machine translation (NMT) models, which is shown in Figure 1(c). However, the model size and capabilities have significantly evolved from NMT models to LLMs, making a comprehensive analysis of inter-layer redundancy in modern LLMs essential. Additionally, SAN requires re-training models from scratch with a complex training strategy to achieve lossless speedup, limiting its applicability to LLMs. In contrast, our method LiSA, as shown in Figure 1(d), can be applied to any existing transformer-based LLMs by only training a few parameters.

2.2 Large Language Models

For modern LLMs, KV cache, which stores history representations, has become an essential technique for accelerating inference. It involves two stages: (1) Prefilling, which initializes the KV cache for each layer; (2) Auto-regressive decoding, which updates the KV cache progressively. However, massive memory and computation consumption are still raised in the inference phase of LLMs (Zhang et al., 2022; Touvron et al., 2023a; AI@Meta, 2024).

2.2.1 Reducing Redundancy Within the Attention Mechanisms of LLMs

Compressing the KV Cache. It is commonly observed that the attention weight matrices are sparse, following a strong power law distribution (Kitaev et al., 2020; Verma, 2021; Choromanski et al., 2021). This indicates that most tokens memorized in the KV cache are redundant. Some works show that only a few fixed tokens greatly catch attention, thus propose to identify and only store these ‘‘important’’ tokens (Liu et al., 2023a; Zhang et al., 2023; Xiao et al., 2024; Ge et al., 2024). Following works continually improve the identification algorithm to reduce performance loss (Adnan et al., 2024; Devoto et al., 2024; Guo et al., 2024). Other studies either store the low-rank representation of tokens (DeepSeek-AI et al., 2024) or quantize the KV cache (Kang et al., 2024). Recently, Cai et al. (2024) and Yang et al. (2024) control the KV cache budget according to different layers' behavior. Other attempts implement the KV cache only at certain layers (Sun

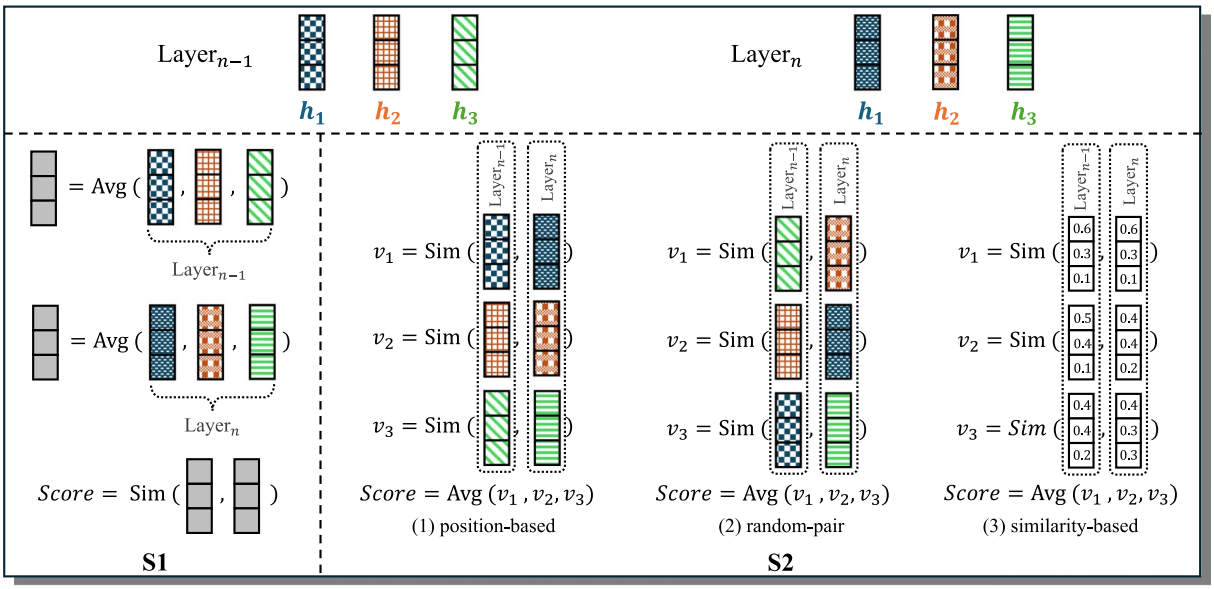


Figure 2: An illustration of strategies for measuring the similarity of attention weights across different layers. The attention mechanism in each layer is assumed to have three heads, represented by blue, red, and green colors, corresponding to their positions within the attention weight matrices. We propose two similarity calculation settings: **S1** computes the average attention weights across heads first and then calculates similarity via Sim(·), e.g., JS divergence; **S2** calculates pairwise similarity scores between aligned heads individually and then averages the scores. Specifically, three head-alignment strategies are considered: (1) *Position-based* alignment matches heads according to their positional indices; (2) *Random-pair* alignment randomly matches heads from two layers; (3) *Similarity-based* alignment pairs each head with its most similar counterpart in the preceding layer, without enforcing a strict one-to-one correspondence.

et al., 2024; Liu et al., 2024; Wu and Tu, 2024; Brandon et al., 2024).

Pruning Attention Heads. Modern LLMs have plenty of attention heads, exacerbating the redundancy. To address this, several approaches have been proposed. Multi-query attention, for instance, shares keys and values among attention heads (Ainslie et al., 2023; Shazeer, 2019). Additionally, Liu et al. (2023b) suggest using a contextual sparsity predictor to identify and dynamically prune unused heads during inference, while Agarwal et al. (2024) propose combining heads based on their similar attention weights.

Indeed, the above methods mainly focus on reducing the redundancy within one component of the attention mechanism. However, analyzing the inter-layer redundancy of the attention mechanism in LLMs is overlooked. Furthermore, although several works of early existing and layer skipping reduce the layer-wise redundancy by pruning entire layers (Gromov et al., 2025; Fan et al., 2025), the after-pruned models struggle with challenging reasoning tasks (Men et al., 2025), leaving the possibility of addressing the inter-layer redundancy within the attention mechanism.

3 Layer-wise Similarity of Attention Weights

Self-attention in Transformer models is essentially a procedure that fuses the information from the context to facilitate better understanding (Xiao and Zhu, 2023). We envision that the attention mechanism of most layers in LLMs may consistently highlight several fixed tokens and assign similar weights to them. To investigate this, *we thoroughly analyze the similarity of attention weights across different layers of LLMs*. Since in MHA the attention mechanism computes separate attention weights for each head, we measure the similarity under the following two settings, as illustrated in Figure 2.

S1: Average attention heads first, then compute similarity. To analyze the similarity of the overall attention scores across layers, we first average the weights of all attention heads within each layer and then compare these weights across all layers.

S2: Align attention heads, compute similarity, and then average the similarity scores. To measure the similarity while considering the diversity of attention heads, one should align heads from two layers ahead, and then compute the average similarity scores. Specifically, we employ three strategies: (1) Position-based alignment matches attention heads according to their respective positions within the attention weight matrices; for example, the head at dimension 0 in one layer is matched with the head at dimension 0 in another layer. (2) Random-pair alignment matches heads from two layers randomly. (3) Similarity-based alignment pairs each head with the most similar counterpart in another layer without enforcing a strict one-to-one correspondence, which reflects the oracle similarity. We envision that similarity-based aligning heads is necessary for achieving high inter-layer similarity, whereas position-based or random-pair alignment hinders the preservation of similarity.

3.1 Experiment Settings

Models and Datasets. We conducted comprehensive experiments on 4 LLMs, specifically LLaMA2-7B (Touvron et al., 2023b), Gemma-7B (Mesnard et al., 2024), LLaMA3-8B (AI@Meta, 2024), and LLaMA3-70B. Our analysis focused on the behavior of the attention mechanism across various tasks, including physical commonsense QA, short sentence translation, coreference resolution, and mathematical reasoning. The corresponding datasets are PIQA (Bisk et al., 2020), WMT (Kocmi et al., 2024), WEC (Eirew et al., 2021), and GSM8K (Cobbe et al., 2021). Among them, WMT consists of 100 human-selected short sentences, while the others contain 100 randomly sampled instances. The first two tasks involve short inputs where capturing local dependencies is sufficient, while the latter two require handling long-range dependencies.[1]

Details of Assessing Attention Similarity. We primarily focus on evaluating the similarity of attention weights from the last input token to all other tokens. Specifically, for each benchmark, we feed 100 samples into the model, extract and aggregate attention weight distributions across heads (i.e., **S1** and **S2** illustrated in Figure 2), and compute the Jensen–Shannon (JS) divergence averaged over 100 samples for each pair of layers. For baselines, we report JS scores between the attention weights of two distinct sentences, as shown in Figure 4(a). Moreover, although our main analysis considers the attention weights from the last input token to all other tokens, the experimental results in Table 11 indicate that the attention distributions from the token generated at step 1,024 to all preceding tokens also exhibit strong inter-layer similarity.

3.2 Results

The similarity scores calculated under setting **S1** are shown in Figure 3. Besides, Figure 5(a) records the similarity scores for sub-modules within the attention mechanism. From these results, we get the following observations:

Attention Weights are Remarkably Similar across Transformer Layers, Especially the Ones in Adjacent Layers. We see, first of all, most JS divergence scores sustained at a degree lower around 0.1, indicating that most layers prefer a similar attention pattern regardless of models and tasks. Another interesting finding is that the JS divergence score near the diagonal line remains below 0.05,[2] demonstrating an extremely similar attention pattern in adjacent layers. This is reasonable because adjacent layers' representations are more similar than non-adjacent ones in deep transformer models (Phang et al., 2021). In addition to JS divergence scores, further evidence, such as the top-5 tokens receiving the highest attention weights per layer, presented in Appendix A.1, also supports the high similarity of attention weights across layers.

Few Layers Maintain Different Attention Patterns. Although most layer pairs in LLaMA3-8B exhibit highly similar attention patterns, some layers compute notably different ones. As shown by the white lines in Figure 3, attention weights in layers 1–2 and 11–14 differ from those in other layers. However, the red square at the intersection of the white cross indicates that the attention patterns among these adjacent layers remain internally consistent. Notably, the distribution of such distinct layers

[1] Each input in PIQA and WMT averages approximately 40 tokens, while WEC contains 120 tokens and GSM8K has 450 tokens per instance.

[2] See Figure 12 for the visualization of attention probability distribution pairs with various JS divergence scores.

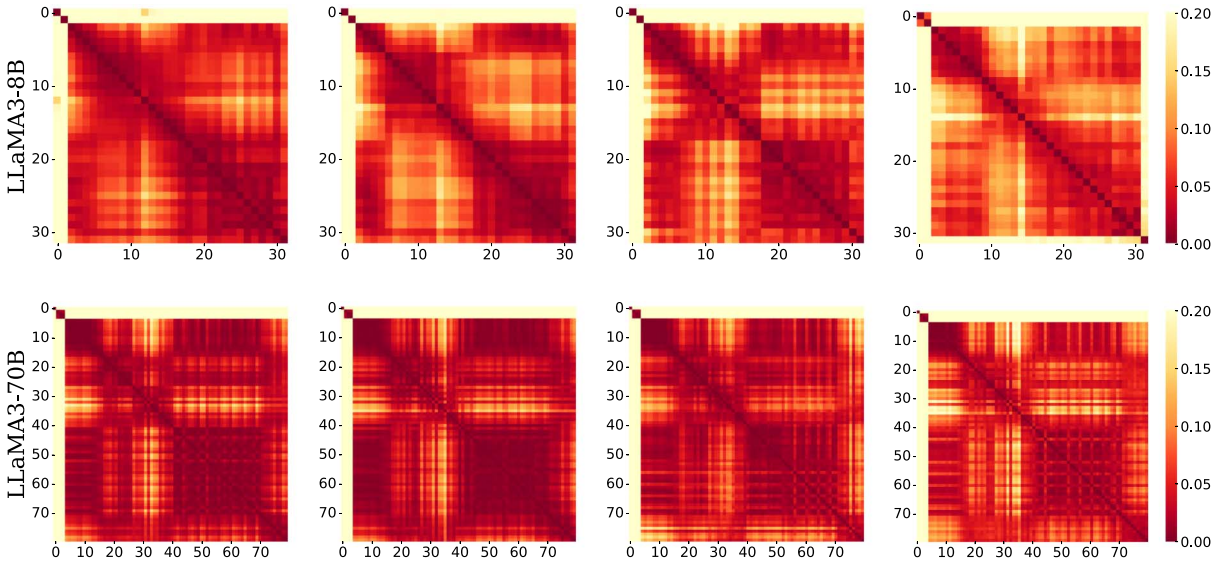


Figure 3: The JS divergence scores for the attention weights between every pair of layers, calculated under setting **S1**. For all figures, both the horizontal and vertical coordinates stand for layer indices. A deeper red color indicates a lower JS divergence score, corresponding to higher similarity. For instance, the cell located at the third row and fourth column of the top-left figure indicates that the JS divergence between the third and fourth layers of LLaMA3-8B is less than 0.05. See Figure 11 for results of LLaMA2-7B and Gemma-7B.

varies across models. For example, Figure 11 presents results for Gemma-9B, where the first four layers differ from the rest, while the remaining 24 layers maintain highly similar attention patterns.

The Similarity of Inter-Layer Attention Weights is Independent of the Task and Reflects an Inherent Property of the Model. Taking LLaMA3-8B as an example, we observe that the similarity between the attention weights of the first layer and the other layers consistently remains low across different tasks. In contrast, the similarity between the fifth and sixth layers is consistently high. This finding is particularly valuable as it allows for the reuse of attention patterns across specific layers, regardless of the task.

Ablation on the First Token. Xiao et al. (2024) observe that the first token in the sequence often receives disproportionately high attention weights. They argue that the model allocates the excess attention mass to the initial token. We also assess the attention similarity by excluding the weights on the first token, re-softmaxing the remaining weights, and computing the JS divergence scores. The experimental results are shown in Figure 4(b), which is comparable to the first

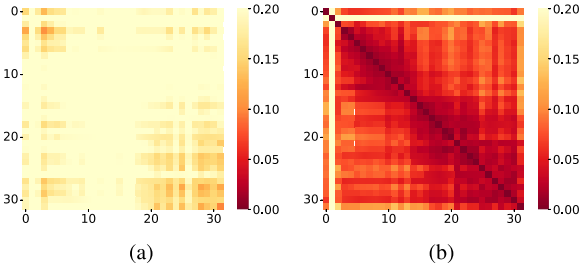


Figure 4: Figure (a) presents the JS divergence between the attention distributions of two distinct sentences, each with an equal number of tokens, across all pairs of layers in LLaMA3-8B. Figure (b) shows the JS divergence of attention weights for LLaMA3-8B on the PIQA dataset, excluding the first token, which is a special token that receives the majority of the attention.

one in the upper left corner of Figure 3. We can see that the JS divergence scores near the diagonal are around 0.05, consistently demonstrating significant similarity of the attention weights in adjacent layers.

Only Attention Weights Exhibit Cross-Layer Similarity. We also measure the similarity of intermediate hidden states in the attention mechanism across each pair of adjacent layers by calculating the cosine similarity. Figure 5(a) shows that only the similarity of the attention scores QK^T (the blue line) remains consistently

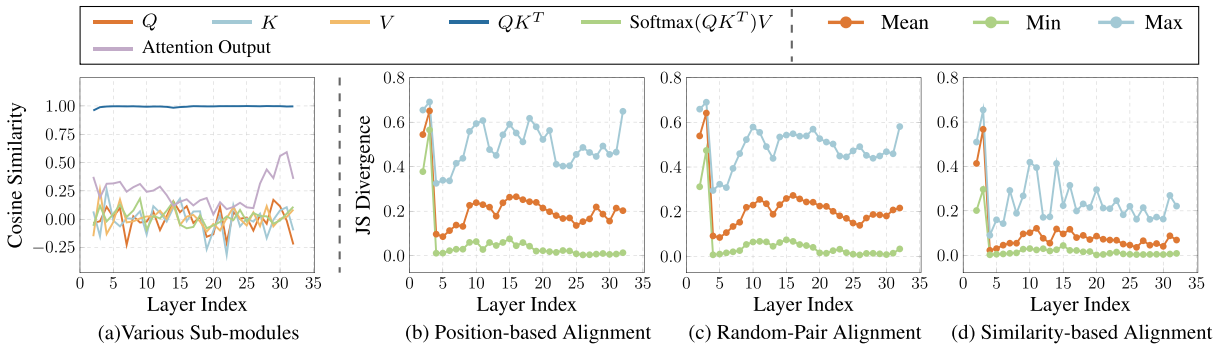


Figure 5: Figure (a) displays the cosine similarity scores for sub-modules within the attention mechanism across each pair of adjacent layers. Figures (b), (c), and (d) present the average JS divergence of attention weights between adjacent layers under three different alignment strategies in setting **S2**: position-based, random-pair, and similarity-based, respectively. Lines are added to improve the visual clarity of trends between discrete layers, even though the x-axis represents discrete layer indices.

close to 1 across layers, while the cosine similarity of other intermediate hidden states stays around 0. This indicates that, while most Transformer layers maintain similar attention patterns, they still serve distinct functions, as their Q, K, and V matrices capture different features. This reflects that these models learn implicit attention patterns across layers while maintaining distinct representations within each layer.

We further analyze the similarity of attention weight while considering the diversity of attention heads, i.e., calculating similarity scores under setting **S2**. Experimental results of LLaMA3-8B on GSM8K are shown in Figure 5(b), (c), and (d).

Similarity score falls when attention heads are matched based on positions. As shown in Figure 5(b), the mean values of JS divergence are around 0.2, indicating that an attention head in the current layer is not always similar to the one at the same position in the shared attention matrix. We attribute this to the fact that the parameters do not have an inherent positional relationship in neural networks. Thus, position-based alignment is equivalent to random-pair alignment, which is demonstrated by the similar results between Figure 5(b) and (c).

Aligning with the most similar head recovers the similarity. We further measure the oracle similarity by aligning the most similar head for the one in the current layer and calculating the average similarity. From Figure 5(d), we see the similarity scores remain below 0.1 in most layers, which indicates that most attention heads can be aligned with a highly similar one in other layers. It also implies that directly utilizing the shared attention weight matrix might be sub-optimal, and it is crucial to align attention heads beforehand.

4 Sensitivity to Attention Weights

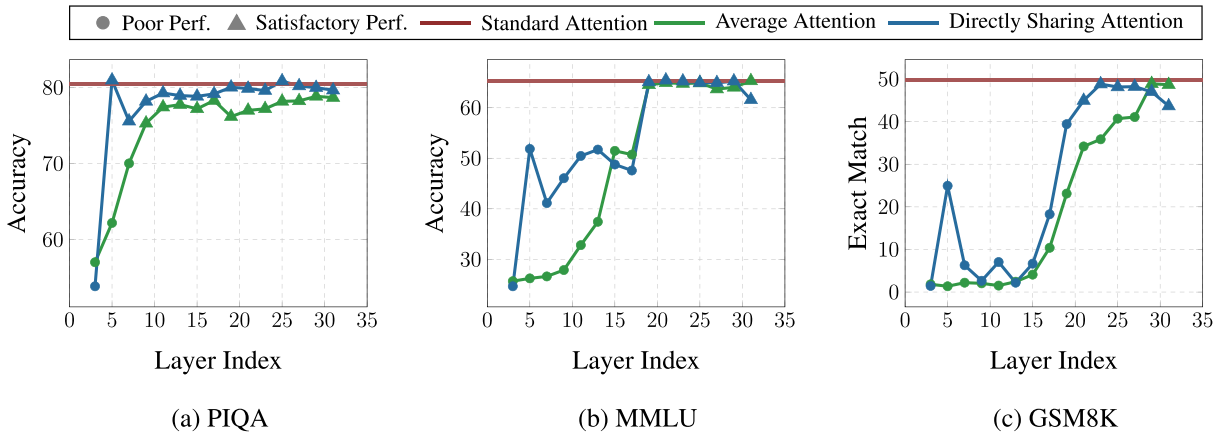
Although the attention weights across different LLM layers are highly similar, they are not identical, and sharing them still introduces minor deviations in the model. Previous studies have shown that even small distortions in parameters or embeddings during the inference process of LLMs can lead to notable performance degradation (Ma et al., 2025; Zhao et al., 2023). Therefore, the next step is to *analyze how deviations in the attention weights affect performance*.

Here, we select two corrupted attention patterns to simulate deviations from the standard attention weights. The first pattern is the attention weight matrix of the front layer without alignment, i.e., directly sharing weights, as depicted in Figure 1(c). Moreover, inspired by AAN (Zhang et al., 2018), the second pattern assigns a uniform attention score across all token positions, i.e., the average weights $\frac{1}{l}$, illustrated in Figure 1(b).

Subsequently, to assess sensitivity, we apply the two aforementioned patterns to successive pairs of adjacent layers (e.g., layers 3–4, 5–6, 7–8). Under this setup, a significant performance drop from the original model indicates that these layers are particularly sensitive to deviations in attention weights. The findings guide the design of the attention-sharing configurations detailed in Section 5.1.

4.1 Results

We conducted experiments on three datasets, including PIQA, MMLU (Hendrycks et al., 2021),



(a) PIQA (b) MMLU (c) GSM8K

Figure 6: The performance of LLaMA3-8B when applying the average and directly sharing attention strategies to successive pairs of adjacent layers (e.g., layers 3–4, 5–6, 7–8). For instance, the leftmost two points in Figure (a) show the performance when DS and average attention are applied to layers 3 and 4 of the model. The red line indicates the model's original performance, in which standard attention is retained across all layers. When the performance of an attention pattern achieves 90% of the original model's performance, the point is marked with a triangle; otherwise, it is marked with a circle. See Figure 13 for the results of LLaMA2-7B.

and GSM8K. From Figures 6 and 13, we draw the following conclusions.

Shallow layers are sensitive to the attention score while deep layers are not. We can see that, in shallow layers, relatively small deviations in attention weights, like sharing attention weights, are more likely to cause a performance collapse. On the contrary, even significant changes happening in deep layers, like averaging attention weights, influence the performance inconspicuously. It indicates that the small deviations contain specific features unique to each layer, which are necessary for sharing attention weights.

The sensitivity of layers is task-dependent. To retain 90% of the original performance (represented by the points forming a triangular shape), models need to preserve standard attention in different layers depending on the dataset: the shallow layers for PIQA, the first half of the layers for MMLU, and most layers for GSM8K. Upon further analysis of the datasets, we attribute this phenomenon to varying task difficulty. Specifically, PIQA is a relatively simple two-choice benchmark requiring only basic physical commonsense reasoning. MMLU, by contrast, is a more challenging four-choice benchmark comprising 57 knowledge-intensive tasks. Unlike these two, GSM8K demands step-by-step mathematical reasoning to arrive at the final answer, indicating a significantly higher level of complexity. This trend aligns with findings from early-exit studies, which show that the optimal layer for exiting generation correlates with input difficulty—simpler inputs can be processed by early layers, while more complex inputs necessitate deeper-layer computation (Matsubara et al., 2023).

5 Reducing the Inter-layer Redundancy

Since the attention scores are similar across Transformer layers, it's a natural step to reuse these results across multiple layers, making the inference more efficient. However, this faces two challenges:

- *An alignment of attention heads in two layers is crucial for maintaining high similarity.*
- *For sensitive layers, minor attention weight deviations cause performance collapses.*

We address these challenges by introducing two lightweight components: an *attention heads alignment* module and a *difference compensation* module. The main idea is that we not only align the most similar heads in the shared weights matrix for each head but also compensate for deviations by approximating the difference between the shared weights matrix and the original one. Bring it all together, we present LiSA, which significantly reduces the inter-layer redundancy of attention in well-trained LLMs with minimal loss. Moreover, we theoretically analyze the efficiency of LiSA in Section 5.2.

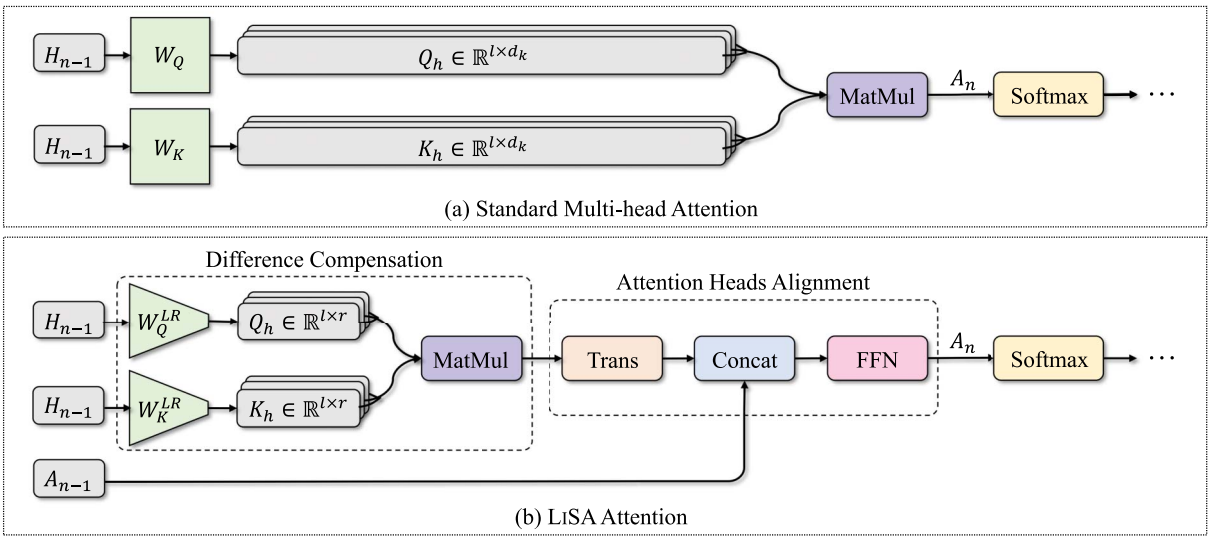


Figure 7: A comparison of LiSA with the standard multi-head attention.

5.1 Methodology

As shown in Figure 7, LiSA reconstructs the calculation steps prior to Softmax$(\cdot)$ in the self-attention mechanism for a better utilization of the shared attention weights.

Learn to Share Attention Weights. Let A denote the attention score matrix, i.e., the intermediate output computed prior to applying Softmax$(\cdot)$. We use A_n to represent the attention score matrix at layer n; for example, A_0 corresponds to the first layer of an LLM. When the layer n arms with LiSA, the attention heads alignment module accepts a matrix A_{n-1} from the adjacent front layer $n-1$ and produces an aligned one A_{n-1}^{align}. Specifically, given a matrix $A \in \mathbb{R}^{h\times l\times l}$, we first transpose it to $A^\top \in \mathbb{R}^{l\times l\times h}$, and then use feed-forward networks (FFNs) to rearrange attention heads and produce the aligned matrix A^{align}.

As illustrated in Figure 8, we take an example to explain how FFNs can align attention heads. For simplicity, we start with a one-layer FFN. Supposing that $h = 3$ and we need to achieve such alignment: $1 \to 3$, $2 \to 2$, and $3 \to 1$. The shared weight matrix can be aligned by multiplying it with a permutation matrix. Moreover, since the weights of the FFN are consecutive, it also performs as fusing the weights from multiple attention heads.

Low-rank Projection Closes Gaps. For the difference compensation module, we first use two low-rank linear projections $W_{LR}^Q, W_{LR}^K \in \mathbb{R}^{d\times r}$ as substitutes for W^Q and W^K. Given the input

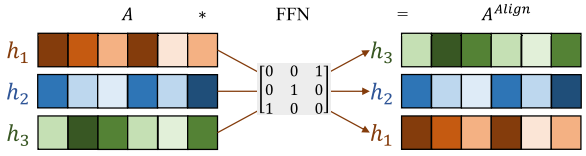


Figure 8: An illustration of how FFNs rearrange the attention heads.

hidden state H, these linear projections are promoted to capture specific features for the current layer. The resulting Q_{LR} and K_{LR} matrices are then processed by the scaled dot-product mechanism to derive the difference $A_\Delta \in \mathbb{R}^{h\times l\times l}$, which is subsequently integrated into the shared attention weight matrix through addition or linear fusion. The whole process is shown as follows:

$$A_\Delta = \frac{HW_{LR}^Q(HW_{LR}^K)^T}{\sqrt{r}}, \tag{3}$$

$$A = \text{Integrate}(A_{n-1}^{align}, A_\Delta). \tag{4}$$

Note that if a tiny dimension r is used, such that $r \ll d$, the representation of Q and K matrices is significantly compressed, thus we can save the memory consumption.

An Overview of LiSA. Complete LiSA is shown in Figure 7. To facilitate more precise alignment, we extend the input of the attention heads alignment module by concatenating $A_{n-1}^\top \in \mathbb{R}^{l\times l\times h}$ with $A_\Delta^\top \in \mathbb{R}^{l\times l\times h}$. Then the attention heads alignment module fuses two matrices and outputs the final attention score matrix A_n. Surprisingly, a super lightweight two-layer

or single-layer FFN performs effectively in this module. See Figure 14 for a well-trained FFN.

Selecting Layers to Implement LiSA. To optimally preserve the performance, we consider the robustness of a layer to variations in attention weights and the similarity of layer-wise attention scores when implementing LiSA.

- Selecting robust layers. Given an LLM, we assess each layer's robustness by evaluating its performance when armed with the same attention-sharing strategy described in Section 4, focusing on challenging mathematical reasoning tasks. For example, Figure 9 suggests layers 21–30 of LLaMA3-8B are preferred to implement LiSA.
- Excluding the first and last few layers. The attention weights in these layers differ from those in adjacent layers, suggesting that attention sharing may not be appropriate for them. For instance, in the upper one of Figure 3(d), low similarity values (i.e., yellow blocks) are located in the first two rows and columns, as well as the last row and column.
- Implementing LiSA at intervals. For layers beyond the first two categories, we recommend alternating LiSA with standard attention—for instance, applying LiSA for two consecutive layers, followed by a standard attention layer, and repeating this pattern. This periodic application of LiSA prevents continuous sharing of the same attention weights, which could limit performance.

Training Strategy. We only train the newly involved parameters, i.e., those of attention heads alignment and difference compensation modules, which further reduce the training cost of LiSA. For instance, only 56 million parameters in LLaMA3-8B (0.7% of total) are trained to apply LiSA to more than half of the layers. Moreover, to achieve efficient uptraining, we leverage the knowledge distillation technique. Aligning with feature-based knowledge methods (Romero et al., 2015; Passalis and Tefas, 2018; Kim et al., 2018), we regard the original model as a teacher and use its attention scores A_n^* as a supervisory signal. Let S be the set of indices corresponding to the layers

Figure 9: The descending order ranked by each layer's robustness. We evaluated models on the 100 items randomly sampled from the GSM8K training set.

that are equipped with LiSA. Then our regression loss function is formulated as follows:

$$\mathcal{L}_{\text{KD}} = \frac{1}{|S|} \sum_{n \in S} \mathcal{L}_\delta(A_n, A_n^*) \tag{5}$$

where $|S|$ denotes the number of layers in the set S and $\mathcal{L}_\delta(\cdot)$ stands for the Huber loss[3] (Huber, 1992). We also uptrain models on the language modeling task. Given the prefix $x_{<t} = \{x_1, x_2, \ldots, x_{t-1}\}$, the corresponding loss function can be expressed by:

$$\mathcal{L}_{\text{LM}} = -\frac{1}{l} \sum_{t=1}^{l} \log p(x_t | x_{<t}). \tag{6}$$

Integrating these optimizing goals by a predefined weight β, then our overall loss function is

$$\mathcal{L} = \beta \mathcal{L}_{\text{KD}} + (1 - \beta) \mathcal{L}_{\text{LM}}. \tag{7}$$

According to the ablation study shown in Table 7, we set β to 0.25 for all experiments.

5.2 Theoretical Efficiency Analysis

The efficiency of LiSA primarily arises from significantly compressing the K cache, which allows for larger batch sizes and longer sequences under reduced memory pressure during generation, thereby enhancing throughput. Here, we theoretically analyze the memory usage during inference. Its generation process comprises two stages for an LLM armed with the KV cache technique. In the **prefilling** stage, the memory saved by compressing K cache in $|S|$ layers with LiSA is $|S| \times h \times l \times (d_k - r) \times 2$ bytes, while storing an

[3] See Appendix C.1 for the complete formulation.

Base Model	#Total Layers	Model Name	#Sharing Layers	Proportion	Specific Layers
LLaMA3-8B	32	DS (17)	17	53.13%	*5,6,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31*
		DS (21)	21	65.63%	*4,5,7,8,10,11,13,14,16,17,19,20,22,23,24,25,26,27,28,29,30*
		DS (27)	27	84.38%	*4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30*
		LiSA (17)	17	53.13%	5,6,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
		LiSA$_{SL}$ (7+10)	17	53.13%	5,6,17,18,19,20,21,*22,23,24,25,26,27,28,29,30,31*
		LiSA (21)	21	65.63%	4,5,7,8,10,11,13,14,16,17,19,20,22,23,24,25,26,27,28,29,30
		LiSA (27)	27	84.38%	4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
LLaMA2-7B	32	DS (17)	17	53.13%	*5,6,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31*
		DS (21)	21	65.63%	*4,5,7,8,10,11,13,14,16,17,19,20,22,23,24,25,26,27,28,29,30*
		LiSA (17)	17	53.13%	5,6,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
		LiSA$_{SL}$ (7+10)	17	53.13%	5,6,17,18,19,20,21,*22,23,24,25,26,27,28,29,30,31*
		LiSA (21)	21	65.63%	4,5,7,8,10,11,13,14,16,17,19,20,22,23,24,25,26,27,28,29,30
LLaMA2-13B	40	DS (22)	22	55.00%	*7,8,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38*
		DS (26)	26	65.00%	*7,8,9,10,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39*
		LiSA (22)	22	55.00%	7,8,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38
		LiSA$_{SL}$ (10+12)	22	55.00%	7,8,19,20,21,22,23,24,25,26,*27,28,29,30,31,32,33,34,35,36,37,38*
		LiSA (26)	26	65.00%	7,8,9,10,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39

Table 1: The configurations of the directly sharing attention (DS) and LiSA models. We report the proportion of layers employing DS or LiSA attention mechanisms relative to the total number of layers. For instance, LLaMA3-8B+LiSA (27) reduces redundant attention calculations within 84% of the total layers. Layers applied DS and LiSA are indicated in *blue* (italicized) and red, respectively. Layer numbering starts from 1.

attention weight matrix requires $h \times l \times l \times 2$ bytes. Therefore, the total memory reduced by LiSA is $h \times l \times (|S| \times (d_k - r) - l) \times 2$ bytes. When the input sequence length l exceeds $|S| \times (d_k - r)$, more memory is consumed. In the **decoding** stage, LiSA continues to compress the K cache as before and introduces a small weight matrix occupying $h \times l \times 2$ bytes. Consequently, the memory reduction by LiSA is $h \times l \times (|S| \times (d_k - r) - 1) \times 2$ bytes. Given that $|S| \times (d_k - r) \gg 1$, LiSA consistently saves memory in this stage.

Indeed, we can avoid additional memory consumption in the prefilling stage by leveraging the original attention mechanism for the initial inference step. To utilize LiSA in subsequent inference steps, one should calculate and store K_{LR} instead of K in the KV cache. The only difference between this decoding strategy and using LiSA throughout all inference steps is that the original attention weights are used in the first inference step. Thus, this approach is lossless, which is also empirically demonstrated in Table 10. We denote this decoding strategy as NF and do not apply it unless stated.

5.3 Experiment Settings

Model Configuration. We selected LLaMA3-8B, LLaMA2-7B, and LLaMA2-13B as the base models. The first two models, LLaMA3-8B and LLaMA2-7B, each comprise 32 layers with 32 attention heads of 128 dimensions each. In contrast, LLaMA2-13B consists of 40 layers, also with 32 attention heads of 128 dimensions each. We designed several layer-wise sharing configurations, detailed in Table 1. Specifically, LiSA denotes the default structure that the attention heads alignment module uses a two-layer FFN along with `ReLU` as the activation function. This model compresses Q, K by $6\times$ (i.e., $r = 20$ compared to $d_k = 128$). While LiSA$_{SL}$ stands for another structure that leverages a one-layer FFN for alignment and compresses the Q, K by $4\times$ (i.e., $r = 32$ compared to $d_k = 128$). Additionally, directly sharing attention is applied to deep layers.

For the baselines, we report the performance of directly sharing attention (DS) and its uptrained version (DS$_{LoRA}$). Specifically, we use the LoRA training method (Hu et al., 2022) to ensure the number of learnable parameters matches that of LiSA. All trainable models are trained on 4.2 billion tokens, which is a subset of RedPajama-1T.[4] Other setups are reported in Appendix C.1.

Performance Evaluation. Following LLaMA2 and LLaMA3, we conducted extensive evaluations on 13 typical downstream benchmarks. We reported the 0-shot accuracy on PIQA, BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al.,

[4] `https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T`.

Model	Trained Param. (%)	Saved Param. (%)	Compressing Q (times)	Compressing K (times)	Commonsense & Reading Comprehension				
					PIQA	BoolQ	WinoGrande	CoQA	OBQA (5)
LLaMA3-8B	–	–	–	4×	80.69	81.13	73.40	67.40	46.60
DS (17)	–	4.44	all	all	68.61	75.72	65.19	12.67	30.20
DS$_{LoRA}$ (17)	0.70	4.44	all	all	79.82	80.15	72.06	0.00	45.00
DS$_{LoRA}$ (21)	0.86	5.48	all	all	77.80	72.94	63.69	0.00	43.60
DS$_{LoRA}$ (27)	1.11	7.05	all	all	73.83	58.13	53.35	0.00	36.20
LiSA (17)	0.70	3.74	6×	24×	79.87	81.65	73.95	63.53	46.20
LiSA$_{SL}$ (7+10)	0.46	3.98	[4×, all]	[16×, all]	80.63	79.17	73.32	64.90	43.80
LiSA (21)	0.86	4.62	6×	24×	80.14	78.78	72.14	61.52	46.20
LiSA (27)	1.11	5.94	6×	24×	80.69	77.86	70.17	60.23	46.80
LLaMA2-7B	–	–	–	–	79.11	77.74	68.98	63.88	42.60
DS (17)	–	8.47	all	all	62.08	64.89	60.69	1.00	26.60
DS$_{LoRA}$ (17)	1.33	8.47	all	all	77.20	76.39	66.54	0.00	40.00
DS$_{LoRA}$ (21)	1.64	10.46	all	all	75.57	67.92	59.75	0.00	41.40
LiSA (17)	1.33	7.14	6×	6×	78.84	76.79	74.51	60.58	45.80
LiSA$_{SL}$ (7+10)	0.87	6.37	[4×, all]	[4×, all]	78.02	76.67	68.11	61.33	41.00
LiSA (21)	1.64	8.82	6×	6×	78.62	73.24	68.27	52.33	41.40
LLaMA2-13B	–	–	–	–	80.52	80.55	72.14	66.37	49.00
DS (22)	–	8.77	all	all	64.53	70.46	60.14	3.17	30.60
DS$_{LoRA}$ (22)	1.37	8.77	all	all	77.91	77.83	69.30	0.00	48.20
DS$_{LoRA}$ (26)	1.62	10.37	all	all	78.45	76.12	68.11	0.00	44.00
LiSA (22)	1.37	7.40	6×	6×	79.54	80.31	71.74	66.97	50.40
LiSA$_{SL}$ (10+12)	0.99	8.15	[4×, all]	[4×, all]	79.38	80.67	71.74	65.42	50.60
LiSA (26)	1.62	8.74	6×	6×	79.82	78.96	72.14	63.13	48.60

Model	Continued			World Knowledge			Reasoning	LM	Avg. Perf. Preserved (%)
	ARC-E	ARC-C (25)	HellaSwag (10)	TriviaQA	NQ (5)	MMLU (5)	GSM8K CoT (8)	LAMBADA	
LLaMA3-8B	77.61	59.30	82.26	63.39	29.14	64.98	51.71	3.48	–
DS (17)	41.04	29.86	50.00	0.73	1.11	23.96	1.74	936.10	**46.67**
DS$_{LoRA}$ (17)	75.38	56.66	78.44	4.80	0.08	49.14	25.85	6.28	**67.83**
DS$_{LoRA}$ (21)	67.97	47.27	73.37	4.41	0.91	35.16	4.85	7.86	**58.06**
DS$_{LoRA}$ (27)	54.21	28.07	57.88	3.81	0.03	23.85	0.38	22.47	**45.38**
LiSA (17)	79.29	58.96	81.17	57.66	27.17	61.22	45.94	3.79	**96.77**
LiSA$_{SL}$ (7+10)	77.74	59.04	79.85	53.11	25.01	61.69	42.76	4.89	**94.31**
LiSA (21)	74.28	55.12	80.83	52.38	26.04	59.52	39.27	3.96	**92.63**
LiSA (27)	74.92	53.33	79.43	43.65	25.65	50.58	31.77	4.37	**88.38**
LLaMA2-7B	74.58	53.24	78.59	52.54	26.01	45.94	14.18	3.76	–
DS (17)	36.66	29.52	35.82	0.11	0.39	1.85	0.53	20594.64	**39.47**
DS$_{LoRA}$ (17)	63.05	46.33	72.85	3.93	0.00	35.19	6.14	8.28	**64.82**
DS$_{LoRA}$ (21)	62.63	42.32	67.90	3.64	0.91	25.32	2.05	8.91	**58.05**
LiSA (17)	71.09	51.62	76.96	50.93	21.94	43.83	12.96	4.26	**97.26**
LiSA$_{SL}$ (7+10)	71.04	51.19	76.03	39.10	17.89	42.05	8.26	5.20	**89.11**
LiSA (21)	71.17	50.26	75.49	39.01	17.51	35.37	10.24	4.71	**87.36**
LLaMA2-13B	77.48	59.73	82.50	60.86	29.81	50.51	24.03	3.37	–
DS (22)	39.44	29.86	44.44	0.28	0.42	43.57	0.00	12877.52	**46.76**
DS$_{LoRA}$ (22)	69.95	51.79	77.93	0.36	1.50	47.16	7.96	4.89	**65.95**
DS$_{LoRA}$ (26)	67.51	49.23	75.96	0.30	0.71	42.99	6.14	5.79	**62.61**
LiSA (22)	74.49	57.17	81.20	55.67	26.45	48.75	21.30	3.74	**96.44**
LiSA$_{SL}$ (10+12)	72.85	56.48	80.77	39.41	24.63	50.05	19.03	3.97	**92.68**
LiSA (26)	74.54	54.95	80.81	36.62	24.63	48.09	15.85	3.87	**90.13**

Table 2: Performance on 13 typical benchmarks. Columns 2–5 present several attributes of each model. For instance, equipping LLaMA3-8B with LiSA$_{SL}$ (7+10) requires training 0.46% of total parameters while saving 3.98% of them. In this configuration, LiSA compresses the Q and K matrices in the first 7 layers by factors of $4\times$ and $16\times$, respectively, while the remaining 10 layers cut down the Q and K matrices via DS. In the last column, we report the average preserved performance across all benchmarks, excluding LAMBADA. We only report the performance of DS (17) and DS (21) because sharing more layers further impairs overall performance.

2021), and ARC easy (ARC-E) (Bhakthavatsalam et al., 2021); 5-shot accuracy on OBQA (Mihaylov et al., 2018) and MMLU; 10-shot accuracy on HellaSwag (Zellers et al., 2019); and 25-shot accuracy on ARC challenge (ARC-C). For the exact match score, we reported 0-shot performance on TriviaQA (Joshi et al., 2017), 8-shot chain-of-thought (Wei et al., 2022) performance on GSM8K, and 5-shot performance on Natural Questions (NQ) (Kwiatkowski et al., 2019). Furthermore, we included 0-shot extract match score on CoQA (Reddy et al., 2019) and the

[Input, Output]	[128, 512]	[128, 1024]	[512, 128]	[512, 1024]	[512, 3072]	[1024, 1024]	[1024, 3072]	[2048, 512]	Avg. Improv.
LLaMA3-8B	538	395	1597	408	201	416	194	684	–
LiSA (17)	562 +4.4%	427 +8.1%	1669 +4.5%	449 +10.1%	232 +15.9%	461 +10.7%	221 +13.6%	775 +13.2%	**10.1%**
LiSA (21)	582 +8.1%	445 +12.7%	1736 +8.7%	463 +13.5%	241 +20.3%	472 +14.7%	233 +19.9%	789 +15.3%	**14.2%**
LiSA (27)	596 +10.8%	455 +15.3%	1797 +12.5%	483 +18.5%	260 +29.4%	501 +20.5%	247 +27.2%	834 +21.9%	**19.5%**
LiSA $_{SL}$ (7+10)	563 +4.6%	433 +9.7%	1733 +8.6%	455 +11.7%	231 +15.2%	459 +12.6%	225 +12.0%	774 +13.1%	**10.9%**
LLaMA2-7B	875	544	2256	544	200	506	193	727	–
LiSA (17)	1008 +15.2%	645 +18.7%	2520 +11.7%	673 +23.8%	248 +23.9%	553 +9.1%	233 +20.8%	862 +18.6%	**17.7%**
LiSA (21)	1396 +59.5%	683 +25.6%	3062 +35.7%	707 +30.0%	260 +30.0%	653 +29.0%	250 +29.2%	870 +19.7%	**32.3%**
LiSA $_{SL}$ (7+10)	1224 +39.9%	696 +28.0%	2751 +21.9%	605 +11.2%	224 +12.0%	549 +8.4%	209 +8.2%	803 +10.5%	**17.5%**
LLaMA2-13B	710	409	1679	352	140	320	127	460	–
LiSA (22)	997 +40.5%	560 +36.9%	2059 +22.6%	474 +34.7%	187 +33.2%	432 +35.0%	172 +35.9%	549 +19.3%	**32.2%**
LiSA (26)	1043 +46.9%	584 +42.9%	2155 +28.3%	504 +43.2%	198 +41.2%	460 +43.7%	185 +46.0%	591 +28.5%	**40.1%**
LiSA $_{SL}$ (10+12)	896 +26.2%	492 +20.2%	1902 +13.3%	408 +16.0%	158 +12.4%	374 +6.3%	146 +4.2%	532 +15.5%	**14.3%**

Table 3: Throughput (token/s) on a A800 80GB GPU with different systems. ‘‘[128, 512]’’ denotes a prompt length of 128 and a generation length of 512.

perplexity on LAMBADA (Paperno et al., 2016). More details are provided in Appendix C.2.

Efficiency Evaluation. Aligning with Zhang et al. (2023), we evaluated the end-to-end throughput and latency of our system. Throughput is defined as the number of prompted and generated tokens per unit of time, calculated as (prompted tokens + generated tokens)/(prompt time + decoding time). Latency refers to the total time consumed by the whole generation process. We conducted each experiment 10 times and reported the averaged results to ensure reliability and consistency across evaluations. All evaluated models are equipped with the KV cache to speed up generation. Moreover, it is worth noting that LLaMA3-8B serves as a strong baseline, with the GQA technique (Ainslie et al., 2023) compressing the KV cache by $4\times$ compared to MHA.

5.4 Main Results

Performance on Downstream Tasks. Table 2 illustrates that employing LiSA to share attention weights across layers in existing LLMs results in minimal performance loss across various domains and sizes of models. Notably, our best-performing model, LLaMA3-8B+LiSA (17), which implements the LiSA structure in over half of its layers, maintains comparable performance as the original model while adding only a few trainable parameters. Furthermore, despite sharing weights across most layers, LLaMA3-8B+LiSA (27) shows minimal performance degradation on most benchmarks. In comparison, directly sharing attention (DS) leads to significant performance declines. For instance, applying DS to 17 layers in LLaMA3-8B results in the model retaining merely 46.67% of its original performance. Even though uptraining the DS with the same number of parameters and an equivalent amount of data, the significant performance declines observed in the DS$_{LoRA}$ models highlight severe impairments to the original capabilities. These findings underscore LiSA's effectiveness as a robust solution for sharing attention weights across layers in LLMs.

End-to-end inference efficiency evaluation. We first examine the throughput *under the limitation of 80GB memory* on the same A800 GPUs with variable batch sizes. To avoid extra memory consumption, we apply the NF decoding strategy when the length of the input sequence surpasses 2048. Table 3 shows that LiSA achieves significant throughput improvements across a range of input-output scenarios, with increases ranging from 17.5% to 32.3% for LLaMA2-7B. Moreover, when LiSA is implemented on a larger model like LLaMA2-13B, a greater improvement in throughput is observed, highlighting that LiSA's benefits grow as the model scales. It is important to note that LLaMA3-8B serves as a robust baseline, where the GQA technique has compressed the KV cache by $4\times$ compared to MHA. When equipped with LiSA, 10.1% to 19.5% improvements are still observed. Additionally, we report the generation latency *under the same batch size settings* in Table 4, which indicates that LiSA consistently reduces the latency compared to the baseline.

5.5 Pre-training From Scratch

We argue that if heads are explicitly aligned by directly sharing when training an LLM from scratch, then the attention heads alignment module can be discarded, and the predicted difference can

Batch Size	8	8	16	32
[Input, Output]	[2048, 512]	[512, 1024]	[128, 1024]	[128, 512]
LLaMA3-8B	35.43	46.31	61.23	43.37
LiSA (17)	33.69 4.9%	43.26 6.6%	57.57 6.0%	41.04 5.4%
LiSA (21)	33.45 5.6%	42.53 8.2%	56.42 7.9%	39.68 8.5%
LiSA (27)	32.68 7.8%	41.58 10.2%	54.32 11.3%	39.67 8.5%
LiSA$_{SL}$ (7+10)	31.35 11.5%	41.99 9.3%	55.54 9.3%	40.44 6.8%
LLaMA2-7B	32.49	37.72	45.15	27.8
LiSA (17)	28.37 12.7%	34.12 9.5%	40.37 10.6%	25.06 9.9%
LiSA (21)	27.43 15.6%	32.33 14.3%	39.66 12.2%	24.22 12.9%
LiSA$_{SL}$ (7+10)	29.29 9.8%	34.21 9.3%	40.99 9.2%	21.52 22.6%
LLaMA2-13B	48.88	56.25	63.59	35.53
LiSA (22)	41.23 15.7%	51.02 9.3%	57.16 10.1%	30.83 13.2%
LiSA (26)	39.59 19.0%	50.62 10.0%	56.74 10.8%	29.82 16.1%
LiSA$_{SL}$ (10+12)	43.36 11.3%	50.96 9.4%	57.67 9.3%	31.43 11.5%

Table 4: Generation latency (sec) on a A800 80GB GPU with different systems.

Model	OBQA	HellaSwag	PIQA	BoolQ	WinoGrande	ARC-E
MHA	24.20	28.95	58.49	61.47	52.01	35.44
GQA	24.40	28.29	59.03	57.58	50.91	35.65
DS (8)	25.20	27.68	58.71	61.10	52.49	34.18
LiSA$_{plus}$ (8)	26.20	27.92	58.65	62.02	50.20	35.14

Table 5: Performance of different attention models pre-trained from scratch. The original model consists of 12 layers, each with 12 attention heads, and an attention head dimension of 64. The layer-wise sharing configuration for DS (8) and Plus (8) is ''2, 3, 4, 6, 7, 8, 10, 11''. For the GQA model, we set the number of KV attention heads to 2.

be directly added to the shared weight matrix, thus a more concise and efficient LiSA$_{plus}$ will be achieved. To investigate this, we pre-train LLaMA-like models with 12 layers and 164 million parameters on 10 billion tokens.[5] Performance shown in Table 5 demonstrates that both directly sharing weights and applying LiSA$_{plus}$ across two-thirds of total layers are lossless. The evaluation losses are shown in Figure 15. These experimental results not only show the potential of LiSA in the pre-training LLMs from scratch but also mirror our observation of the redundancy within the inter-layer attention mechanism again.

5.6 Task-specific LiSA

For certain vertical-domain tasks, it is preferable to estimate LiSA's performance under a given attention-sharing configuration in a training-free manner. Thus, we can iteratively refine the configuration based on its estimated effectiveness, thereby improving overall results. We show that DS can serve as a substitute to estimate LiSA's performance without training, and then we provide refinement suggestions for different situations.

Specifically, we assess the performance loss of DS under the same attention-sharing configuration as LiSA. Here, DS serves as the lower bound of LiSA. If DS exhibits large performance degradation, LiSA must devote more effort to preserve performance, indicating that the corresponding task is more challenging. As shown in Table 9, we categorize 12 tasks into three levels based on DS's performance loss: Low-loss (green): loss within 0%–40%; Moderate-loss (yellow): within 40%–70%; and High-loss (red): loss within 70%–100%.

After determining the level, we outline several criteria: For low-loss tasks, LiSA performs without loss and even slightly outperforms the original model, allowing us to further improve efficiency by applying attention sharing in more layers; For a moderate-loss task, LiSA can generally achieve near-lossless performance; For a high-loss task, LiSA can still recover most of the performance. However, if stricter performance requirements exist, we can reduce the number of layers employing LiSA to minimize performance loss.

5.7 Ablation Study

We present ablation studies of LiSA on (1) different sequence lengths, (2) instruct-tuned models, and (3) dissecting the effectiveness of each component.

Q_1: Does increasing the number of shots during inference affect LiSA's effectiveness? A_1: No. Table 6 displays the results of incrementally increasing the number of shots. It indicates that LiSA maintains robust performance, effectively leveraging different numbers of shots, similar to the performance of the original model.

Q_2: Does LiSA affect the performance of instruction fine-tuning? A_2: No. We first fine-tune LLaMA3 and our LiSA models on the Alpaca dataset (Taori et al., 2023), and then leverage GPT-4 to judge pairs of responses. The win rate points in Figure 10 show that LiSA models even slightly outperform the baseline.

Q_3: Are all sub-modules have been empirically verified? A_3: Yes. Table 7 presents the results of ablating every sub-module in LiSA, with each model trained on 1 billion tokens. The results highlight the critical roles of the attention

[5]This pre-training corpus takes up 40GB which aligns with GPT-2 (Radford et al., 2019).

Model	BoolQ		PIQA		ARC-E	
	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
LLaMA3-8B	82.26	83.30	82.64	82.86	84.81	84.81
LiSA (17)	83.52	84.31	82.21	82.43	84.30	84.05
LiSA (21)	77.37	77.00	81.28	82.26	82.58	82.62
LiSA (27)	76.15	74.86	81.61	82.21	80.89	82.07

Table 6: Ablation study of different number of shots.

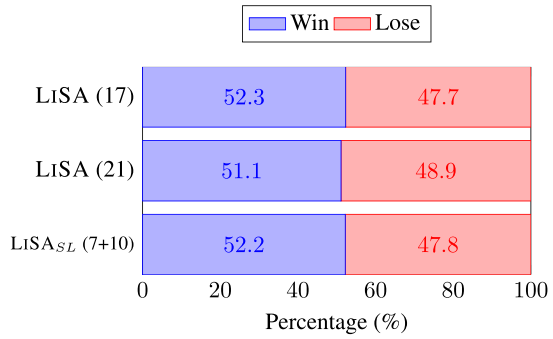


Figure 10: The win rate of LiSA models compared with LLaMA3-8B. All models have been fine-tuned using instruction data.

Model	BoolQ	PIQA	CoQA	MMLU	GSM8K
LLaMA3-8B	81.13	80.69	67.40	65.24	51.71
DS (21)	40.46	56.86	0.11	0.00	2.65
+Align	74.34	77.48	50.28	46.53	7.96
+Diff.	37.86	52.99	0.00	0.61	0.68
+Both	76.85	80.09	63.03	56.78	26.84

Table 7: Ablation study of sub-modules in LiSA. ''+Align'' and ''+Diff.'' mean we individually enable the attention heads alignment and the difference compensation module, respectively. ''+Both'' denotes that we use both modules at the same time.

heads alignment and the difference compensation modules in preserving performance. Preliminary experiments are detailed in Table 8, demonstrating the effectiveness of each setup in LiSA.

6 Conclusion

In this work, we first provide a comprehensive layer-wise redundancy analysis of the attention mechanism in LLMs. We find that: (1) Most Transformer layers perform a highly similar attention pattern; (2) Individual attention heads hinder from directly sharing attention weight; (3) Shallow layers are sensitive to little deviations in attention weight, while deep layers are not. Driven by these insights, we propose a learnable sharing attention mechanism for existing well-trained LLMs. Comprehensive experiments demonstrate that our

Model Configuration		BoolQ	PIQA	CoQA	MMLU	GSM8K
Alignment Structure	Plus	68.72	78.24	48.73	38.38	6.14
	SL	72.81	78.89	56.82	61.58	10.31
	DL + ReLU	**76.85**	**80.09**	**63.03**	**56.78**	**26.84**
	DL + SiLU	75.63	79.38	62.55	57.05	22.30
Hidden Size	128	76.18	79.33	61.63	56.37	24.56
	256	**76.85**	**80.09**	**63.03**	**56.78**	**26.84**
	512	76.48	79.16	61.95	56.30	24.87
Rank of W^Q_{LR}, W^K_{LR}	128	74.65	79.49	60.07	54.62	21.76
	192	76.57	79.49	61.58	56.99	23.65
	320	76.85	80.09	60.03	56.78	26.84
	640	**76.61**	**80.20**	**62.53**	**57.25**	**27.67**
	1024	77.49	79.54	63.07	57.21	30.10
β	**0.25**	**76.85**	**80.09**	**63.03**	**56.78**	**26.84**
	0.50	77.09	79.43	62.78	61.79	24.64
	0.75	75.35	79.00	61.03	61.89	18.35

Table 8: Ablation study of different configurations. Plus indicates the difference matrix is added to the shared attention weight matrix. SL and DL represent one-layer and two-layer FFNs are used in the attention heads alignment module, respectively. The hidden size stands for the intermediate size of the above two-layer FFN. The default configuration denoted as LiSA is **bolded**.

method significantly reduces the inter-layer redundancy of attention, achieving efficient throughput and memory with minimal loss. As far as we know, this is the first attempt to analyze and reduce inter-layer redundancy of attention weights within LLMs. In future work, we plan to investigate whether this problem occurs in large models of other modalities.

Acknowledgments

This work was supported in part by the National Science Foundation of China (nos. 62276056 and U24A20334), the Yunnan Fundamental Research Projects (no. 202401BC070021), the Yunnan Science and Technology Major Project (no. 202502AD080014), and the Program of Introducing Talents of Discipline to Universities, Plan 111 (no. B16009). The authors thank Siming Wu and Peinan Feng for their valuable advice, and extend our sincere gratitude to action editor Xavier Carreras and the anonymous TACL reviewers for their insightful feedback and constructive suggestions.

References

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: KV cache reduction through key tokens selection for

efficient generative inference. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13–16, 2024*. mlsys.org.

Saurabh Agarwal, Bilge Acun, Basil Hosmer, Mostafa Elhoushi, Yejin Lee, Shivaram Venkataraman, Dimitris Papailiopoulos, and Carole-Jean Wu. 2024. CHAI: Clustered head attention for efficient LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21–27, 2024*. OpenReview.net.

AI@Meta. 2024. Llama 3 model card.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6–10, 2023*, pages 4895–4901, Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.298

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16–20, 2020*, pages 268–284, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.19

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. Think you have solved direct-answer question answering? Try ARC-DA, the Direct-Answer AI2 reasoning challenge. *CoRR*, abs/2102.03315.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*, pages 7432–7439, AAAI Press. https://doi.org/10.1609/aaai.v34i05.6239

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. Reducing transformer key-value cache size with cross-layer attention. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10–15, 2024*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069. https://doi.org/10.48550/arXiv.2406.02069

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

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse,

and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Together Computer. 2023. RedPajama: An open source recipe to reproduce LLaMA training dataset.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28–December 9, 2022*. https://doi.org/10.52202/068431-1189

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, Tao Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, and Xiaowen Sun. 2024. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434. https://doi.org/10.48550/arXiv.2405.04434

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28–December 9, 2022*.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. A simple and effective L_2 norm-based strategy for KV cache compression. *arXiv preprint arXiv: 2406.11430*. https://doi.org/10.18653/v1/2024.emnlp-main.1027

Alon Eirew, Arie Cattan, and Ido Dagan. 2021. WEC: Deriving a large-scale cross-document event coreference dataset from Wikipedia. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6–11, 2021*, pages 2498–2510. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.198

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, and Yequan Wang. 2025. Not all layers of LLMs

are necessary during inference. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16–22, 2025*, pages 5083–5091. ijcai.org. https://doi.org/10.24963/ijcai.2025/566

Chao Fang, Aojun Zhou, and Zhongfeng Wang. 2022. An algorithm-hardware co-optimized framework for accelerating N: M sparse transformers. *IEEE Transactions on Very Large Scale Integration Systems*, 30(11):1573–1586. https://doi.org/10.1109/TVLSI.2022.3197282

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: adaptive KV cache compression for LLMs. In the *Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net.

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*, pages 2214–2224.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2020, Online, July 9, 2020*, pages 143–155. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.repl4nlp-1.18

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2025. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24–28, 2025*. OpenReview.net.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. 2024. Attention score is not all you need for token importance indicator in KV cache reduction: Value also matters. *arXiv preprint arXiv:2406.12335*. https://doi.org/10.18653/v1/2024.emnlp-main.1178

Tae Jun Ham, Sungjun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. A^3: Accelerating attention mechanisms in neural networks with approximation. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22–26, 2020*, pages 328–341. IEEE. https://doi.org/10.1109/HPCA47549.2020.00035

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. 2020. ContextNet: Improving convolutional neural networks for automatic speech recognition with global context. In *21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25–29, 2020*, pages 3610–3614. ISCA. https://doi.org/10.21437/Interspeech.2020-2059

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

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, ICLR 2022, Virtual Event, April 25–29, 2022*. OpenReview.net.

Peter J. Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in Statistics: Methodology and Distribution*. Springer, pages 492–518. https://doi.org/10.1007/978-1-4612-4380-9_35

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2022. Perceiver IO: A General architecture for structured inputs & outputs. In the *Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022*. OpenReview.net.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.372

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30–August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1147

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. GEAR: An efficient KV Cache compression recipe for near-lossless generative inference of LLM. *CoRR*, abs/2403.05527. https://doi.org/10.48550/arXiv.2403.05527

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

Jangho Kim, Seonguk Park, and Nojun Kwak. 2018. Paraphrasing complex network: Network compression via factor transfer. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada*, pages 2765–2774.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-BERT: Integer-only BERT quantization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondrej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popovic, Mariya Shmatova, Steinthór Steingrímsson, and Vilém Zouhar. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation, WMT 2024, Miami, FL, USA, November 15–16, 2024*, pages 1–46. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.wmt-1.1

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the*

Association for Computational Linguistics, 7:452–466. https://doi.org/10.1162/TACL_A_00276

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, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval

Yih-Kai Lin, Chu-Fu Wang, Ching-Yu Chang, and Hao-Lun Sun. 2021. An efficient framework for counting pedestrians crossing a line using low-cost devices: the benefits of distilling the knowledge in a neural network. *Multimedia Tools and Applications* , 80(3):4037–4051. https://doi.org/10.1007/S11042-020-09276-9

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

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023a. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10–16, 2023*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. 2023b. Deja Vu: Contextual sparsity for efficient LLMs at inference time. In *International Conference on Machine Learning, ICML 2023, 23–29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*, pages 1412–1421. The Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166

Ziyang Ma, Zuchao Li, Lefei Zhang, Gui-Song Xia, Bo Du, Liangpei Zhang, and Dacheng Tao. 2025. Model hemorrhage and the robustness limits of large language models. *CoRR*, abs/2503.23924. https://doi.org/10.48550/arXiv.2503.23924

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. 2020. LadaBERT: Lightweight adaptation of BERT through hybrid model compression. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8–13, 2020*, pages 3225–3234. International Committee on Computational Linguistics. https://doi.org/10.18653/v1/2020.coling-main.287

Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2023. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):90:1–90:30. https://doi.org/10.1145/3527155

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in large language models are more redundant than you expect. In

Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27–August 1, 2025, pages 20192–20204. Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.findings-acl.1035

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295. https://doi.org/10.48550/arXiv.2403.08295

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31–November 4, 2018*, pages 2381–2391. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1260

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7–12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. https://doi.org/10.18653/v1/P16-1144

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061. PMLR.

Nikolaos Passalis and Anastasios Tefas. 2018. Learning deep representations with probabilistic knowledge transfer. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pages 283–299. Springer. https://doi.org/10.1007/978-3-030-01252-6_17

Jason Phang, Haokun Liu, and Samuel R. Bowman. 2021. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2021, Punta Cana, Dominican Republic, November 11, 2021*, pages 529–538. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.blackboxnlp-1.42

Telmo Pires, António Vilarinho Lopes, Yannick Assogba, and Hendra Setiawan. 2023. One wide feedforward is all you need. In *Proceedings of the Eighth Conference on Machine Translation, WMT 2023, Singapore, December 6–7, 2023*, pages 1031–1044. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.wmt-1.98

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. ZeRO: Memory optimization towards training a trillion parameter models. *CoRR*, abs/1910.02054.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics,*, 7:249–266. https://doi.org/10.1162/TACL_a_00266

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68. https://doi.org/10.1162/TACL_a_00353

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106. https://doi.org/10.1145/3474381

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement Pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, et al. 2022. BLOOM: A 176B-parameter open-access multilingual language model. *CoRR*, abs/2211.05100. https://doi.org/10.48550/arXiv.2211.05100

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

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*, pages 8815–8821. AAAI Press. https://doi.org/10.1609/aaai.v34i05.6409

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-attentional acoustic models. In the *19th Annual Conference of the International Speech Communication Association, Interspeech 2018, Hyderabad, India, September 2–6, 2018*, pages 3723–3727. ISCA. https://doi.org/10.21437/Interspeech.2018-1910

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10–15, 2024*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: A compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020*,

pages 2158–2170. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.195

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2023. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):109:1–109:28. https://doi.org/10.1145/3530811

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4–8, 2016*, pages 2464–2469. IEEE. https://doi.org/10.1109/ICPR.2016.7900006

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971. https://doi.org/10.48550/arXiv.2302.13971

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288. https://doi.org/10.48550/arXiv.2307.09288

Madhusudan Verma. 2021. Revisiting Linformer with a modified self-attention with linear complexity. *CoRR*, abs/2101.10277.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28–August 2, 2019, Volume 1: Long Papers*, pages 5797–5808. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1580

Chenglong Wang, Hang Zhou, Kaiyan Chang, Bei Li, Yongyu Mu, Tong Xiao, Tongran Liu, and JingBo Zhu. 2024. Hybrid alignment training for large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11–16, 2024*, pages 11389–11403. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.676

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020a. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020*, pages 7675–7688. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.686

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020b. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H.

Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28–December 9, 2022*.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed KV cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11–16, 2024*, pages 11175–11188. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.602

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net.

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, Macao, China, August 10–16, 2019*, pages 5292–5298. ijcai.org. https://doi.org/10.24963/ijcai.2019/735

Tong Xiao and Jingbo Zhu. 2023. Introduction to transformers: An NLP perspective. *CoRR*, abs/2311.17633. https://doi.org/10.48550/arXiv.2311.17633

Dongkuan Xu, Subhabrata Mukherjee, Xiaodong Liu, Debadeepta Dey, Wenhui Wang, Xiang Zhang, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Few-shot task-agnostic neural architecture search for distilling large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28–December 9, 2022*.

Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. 2021. NAS-BERT: Task-agnostic and adaptive-size BERT compression with neural architecture search. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14–18, 2021*, pages 1933–1943. ACM. https://doi.org/10.1145/3447548.3467262

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11–16, 2024*, pages 3258–3270. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.195

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28–August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1472

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15–20, 2018, Volume 1: Long Papers*, pages 1789–1798. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1166

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models.

CoRR, abs/2205.01068. https://doi.org/10.48550/arXiv.2205.01068

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10–16, 2023*.

Jun Zhao, Zhihao Zhang, Yide Ma, Qi Zhang, Tao Gui, Luhui Gao, and Xuanjing Huang. 2023. Unveiling A core linguistic region in large language models. *CoRR*, abs/2310.14928. https://doi.org/10.48550/arXiv.2310.14928

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-demos.38

A Extension for Section 3

A.1 Additional Evidence Supporting Attention Similarity

Results on More Models. Figure 11 shows the extended results of Figure 3, i.e., the JS divergence score of LLaMA2-7B and Gemma-7B. It suggests that highly similar attention patterns persist across most layers of different large language models.

Detailed Results of the Top-5 Most Attended Tokens. To provide a more quantitative assessment, Table 11 presents detailed results for the top-5 tokens with the highest attention weights across the 32 layers of LLaMA3-8B. The first two columns (''1'' and ''1024'') indicate that at generation steps 1 and 1024, the attention weights in adjacent layers are highly similar. Moreover, the ''PIQA'' and ''GSM8K'' columns show that while most layers exhibit similar attention patterns, layers 11 to 17 display more divergent behavior, which corresponds to the white cross lines observed in Figure 3.

JS Divergence Score Instances. Figure 12 shows instances of two attention probability distributions and their corresponding JS divergence scores. Our results indicate that the attention probability distributions are highly similar when the JS divergence falls below 0.05.

JS divergence score baseline. To facilitate a more objective comparison of JS divergence scores, we introduce a baseline that compares attention distributions across different sentences of equal length. As shown in Figure 4(b), the attention weights differ significantly between the two unrelated sentences, highlighting the high similarity observed in adjacent layers.

B Extension for Section 4

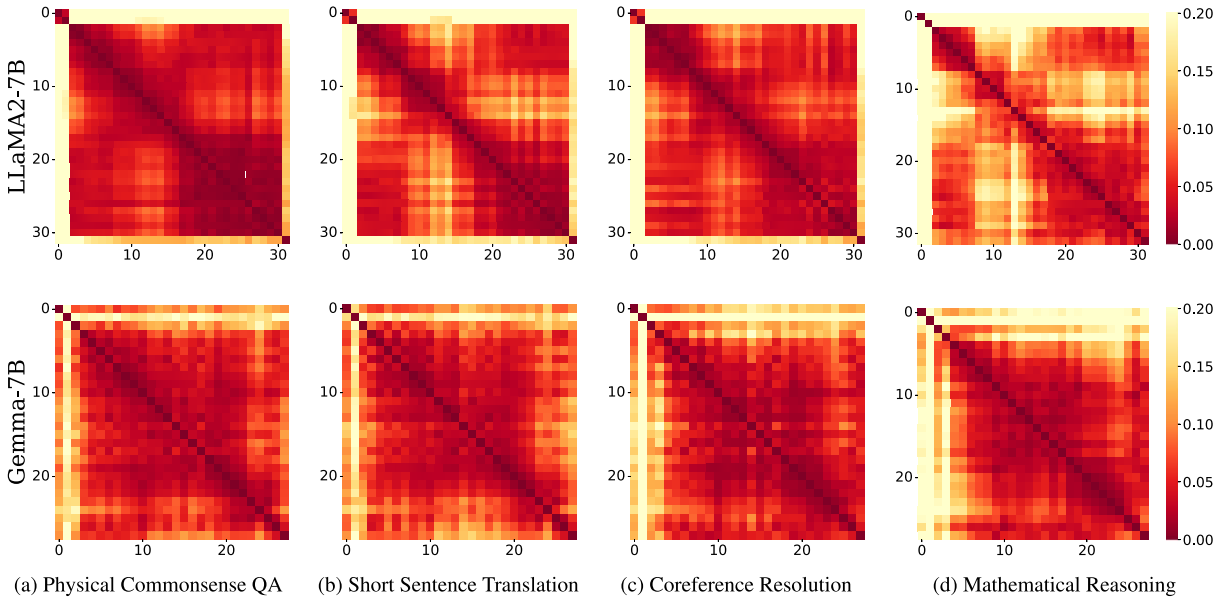
Figure 13 shows the performance of LLaMA2-7B when applying the average and directly sharing attention strategies in every two adjacent layers.

C Extension for Section 5

C.1 Training Setups

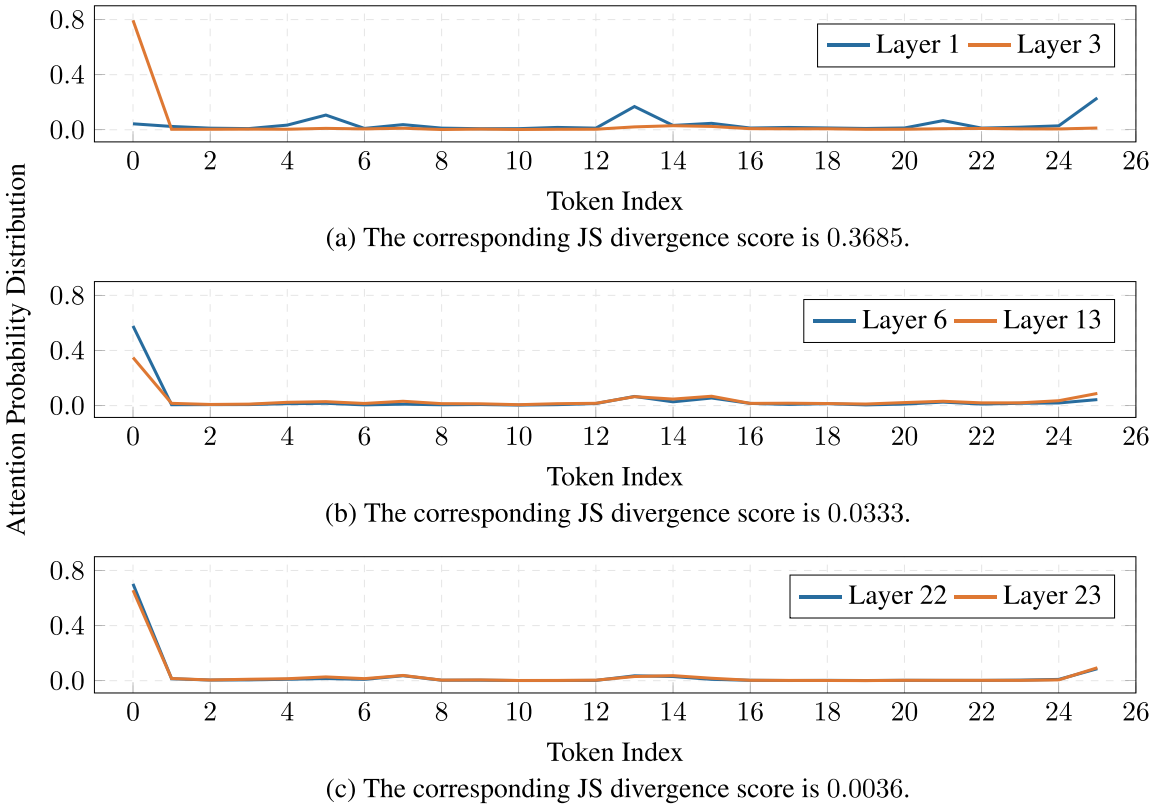
Detailed LiSA Configuration. Both the LLaMA2-7B and LLaMA3-8B models are equipped with 32 attention heads ($h = 32$) per layer and have hidden state dimensions of $d = 4096$. While LLaMA2-13B consists of 40 attention heads ($h = 40$) per layer and has hidden state dimensions of $d = 5120$.

- When a layer is equipped with LiSA, it uses a two-layer FFN, which involves a 64×256 FFN, a `ReLU` activation function, and a 256×32 FFN. Additionally, LiSA includes two low-rank linear projections: for the LLaMA3-8B model, these projections are $W_{LR}^{Q} \in \mathbb{R}^{4096\times640}$ and $W_{LR}^{K} \in \mathbb{R}^{4096\times160}$; for the LLaMA2-7B, $W_{LR}^{Q}, W_{LR}^{K} \in \mathbb{R}^{4096\times640}$; and for the LLaMA2-13B, $W_{LR}^{Q}, W_{LR}^{K} \in \mathbb{R}^{5120\times800}$ each.
- Besides, LiSA$_{SL}$ uses a one-layer FFN sized 64×32, paired with two low-rank linear projections: for LLaMA3-8B, $WLR^{Q} \in \mathbb{R}^{4096\times1024}$ and $W_{LR}^{K} \in \mathbb{R}^{4096\times256}$; for LLaMA2-7B, both projections are $W_{LR}^{Q}, W_{LR}^{K} \in \mathbb{R}^{4096\times1024}$; and for



(a) Physical Commonsense QA (b) Short Sentence Translation (c) Coreference Resolution (d) Mathematical Reasoning

Figure 11: The JS divergence scores of the attention weights for every pair of layers (calculated under setting S1). The greater the redness, the higher the similarity.



(a) The corresponding JS divergence score is 0.3685.

(b) The corresponding JS divergence score is 0.0333.

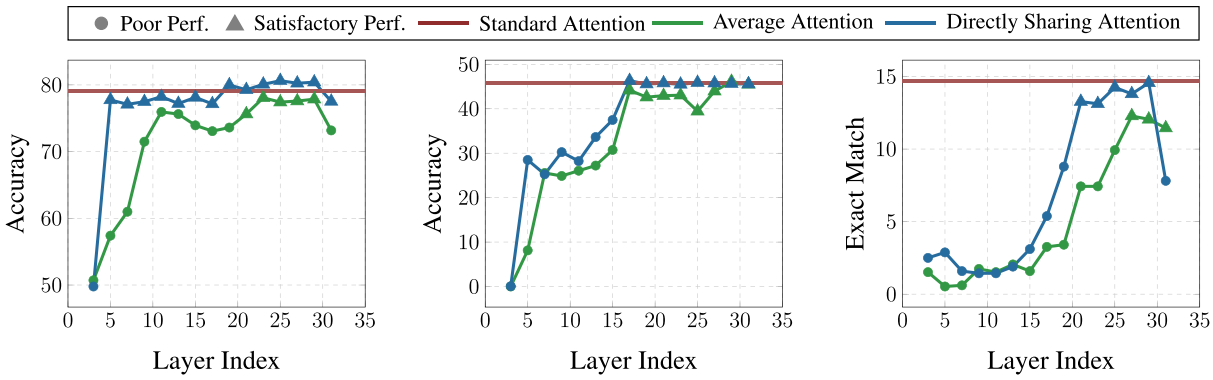
(c) The corresponding JS divergence score is 0.0036.

Figure 12: A visualization of the attention probability distribution in two layers. We also report the corresponding JS divergence score. The horizontal coordinates stand for tokens with different positions.

LLaMA2-13B, $W^Q_{LR}, W^K_{LR} \in \mathbb{R}^{5120\times1280}$ each.

Huber Loss Function. The standard function of Huber loss (Huber, 1992) can be expressed as follows:

$$\mathcal{L}_\delta(a,b) = \begin{cases} \frac{1}{2}(a-b)^2 & \text{if } |a-b| \le \delta \\ \delta(|a-b| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (8)$$



(a) PIQA (b) MMLU (c) GSM8K

Figure 13: The performance of LLaMA2-7B when applying the average and directly sharing attention strategies in every two adjacent layers.

where δ is always set to 1 in our experiments. Indeed, it is a combination of *mean absolute error (MAE)* and *mean squared error (MSE)* loss, which can make the training process more robust.

Datasets. Since the trainable parameters introduced by LISA, only account for 0.46% to 1.64% of the total parameters, we do not need a large training dataset. To obtain high-quality pre-training data, we applied different sampling proportions to subsets of `RedPajama-Data-1T` (Computer, 2023), including 10% of ArXiv, 2% of C4, 100% of StackExchange, 100% of Wikipedia, and 10% of GitHub. The resulting dataset contains 20 billion tokens and we sampled 4.2 and 10 billion tokens from this dataset for the experiments of uptraining and pre-training from scratch.

Main Experiment. We trained all models using the `LLaMA-Factory`.[6] package (Zheng et al., 2024) During the pre-training stage, we set the global batch size to 128, β to 0.25, weight decay to 0.1, number of training epochs to 1, warmup steps to 1500, maximum text length to 1024, and the learning rate to 0.0003. The training process consisted of 40,000 update steps. Additionally, we used DeepSpeed ZeRO-2 (Rajbhandari et al., 2019). All experiments were conducted on eight A800 GPUs.

Preliminary Experiment. To accelerate the training, we trained all models on 1 billion tokens from the `RedPajama-Data-1T-Sample` dataset. Other hyperparameters remain the same as in the main experiment, except for the global batch size, which is set to 16.

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

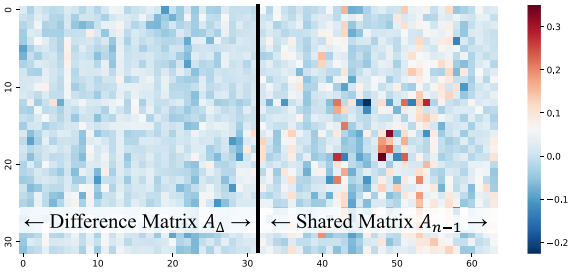


Figure 14: A weight visualization of a well-trained single-layer FFN for aligning attention heads, whose shape is 64×32, i.e., $2h \times h$. Values in two square matrices represent the learned weights accounting for the difference matrix A_Δ and the shared attention weight matrix A_{n-1}, respectively.

C.2 Evaluation Setups

Downstream Tasks. We used the `lm-evaluation-harness` package (Gao et al., 2023) to evaluate the quality of outputs from different models. Except for the number of shots, which is set according to the configurations used by LLaMA2, LLaMA3, and Xia et al. (2024), we kept other hyperparameters at their default settings in the `lm-evaluation-harness` package.

Supervised Fine-tuning. To evaluate the capabilities of following instructions, we first fine-tuned the models on the Alpaca dataset, which contains 52,000 instances. Then, we prompted the models to generate responses on the Alpaca-Eval (Li et al., 2023) data and leveraged GPT-4 (`gpt-4-0613`) to determine which of the two

Benchmark	BoolQ	WinoGrande	PIQA	OBQA	HellaSwag	ARC-E	ARC-C	MMLU	CoQA	NQ	GSM8K	TriviaQA
LLaMA3-8B	81.13	73.40	80.69	46.60	82.26	77.61	59.30	64.98	67.40	29.14	51.71	63.39
DS (17)	75.72	65.19	68.61	30.20	50.00	41.04	29.86	23.96	12.67	1.11	1.74	0.73
Perf. Loss†	***6.67%***	***11.19%***	***14.97%***	***35.19%***	***39.22%***	***47.12%***	***49.65%***	***63.13%***	***81.20%***	***96.19%***	***96.64%***	***98.85%***
LiSA (17)	81.65	73.95	79.87	46.20	81.17	79.29	58.96	61.22	63.53	27.17	45.94	57.66
Perf. Loss	***−0.64%***	***−0.75%***	***1.02%***	***0.86%***	***1.33%***	***−2.16%***	***0.57%***	***5.79%***	***5.74%***	***6.76%***	***11.16%***	***9.04%***
Benchmark	**WinoGrande**	**PIQA**	**OBQA**	**BoolQ**	**ARC-E**	**ARC-C**	**HellaSwag**	**GSM8K**	**TriviaQA**	**NQ**	**MMLU**	**CoQA**
LLaMA3-8B	73.40	80.69	46.60	81.13	77.61	59.30	82.26	51.71	63.39	29.14	64.98	67.40
DS (27)	51.54	56.58	25.40	38.07	30.64	22.78	28.31	2.12	0.04	0.03	0.00	0.00
Perf. Loss†	***29.78%***	***29.88%***	***45.49%***	***53.08%***	***60.52%***	***61.59%***	***65.58%***	***95.90%***	***99.94%***	***99.90%***	***100.00%***	***100.00%***
LiSA (27)	70.17	80.69	46.80	77.86	74.92	53.33	79.43	31.77	43.65	25.65	50.58	60.23
Perf. Loss	***4.40%***	***0.00%***	***−0.43%***	***4.03%***	***3.47%***	***10.07%***	***3.44%***	***38.56%***	***31.14%***	***11.98%***	***22.16%***	***10.64%***

Table 9: The performance loss of DS and LiSA on LLaMA3-8B. We report the performance loss as a percentage in the ''Perf. Loss'' lines. The symbol †represents that the 12 benchmarks are ordered by increasing the performance loss of DS models, from left to right. Green indicates that this is a low-loss task, with the performance loss of DS models below 40%. While yellow denotes that it is a moderate-loss task, with the performance loss of DS models in the range of 40% to 70%. Additionally, red stands for high-loss tasks, with the performance loss of DS models above 70%.

Model	GSM8K	MMLU
	LLaMA3-8B	
LiSA (17)	45.94	61.22
+ NF	47.31	61.22
LiSA$_{SL}$ (7+10)	42.76	61.69
+ NF	41.55	61.69
LiSA (21)	39.27	59.52
+ NF	42.99	59.52
LiSA (27)	31.77	50.58
+ NF	35.10	50.62
	LLaMA2-7B	
LiSA (17)	12.96	43.83
+ NF	12.96	43.24
LiSA$_{SL}$ (7+10)	8.26	42.05
+ NF	7.96	43.18
LiSA (21)	10.24	35.37
+ NF	10.69	35.57

Table 10: Experiment of ablating the NF decoding strategy.

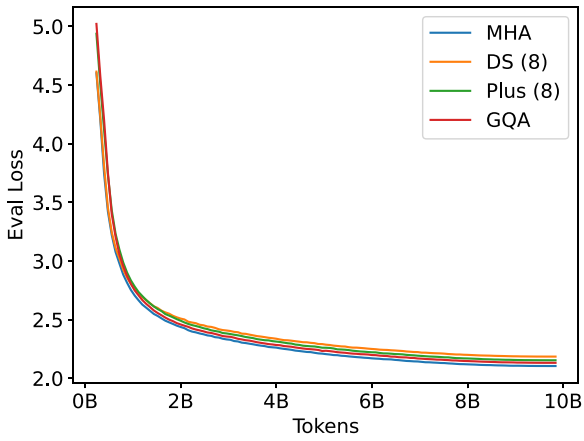


Figure 15: Evaluation loss curves for pre-training LLaMA-like models with various attention mechanisms. The original model consists of 12 layers, each with 12 attention heads, and an attention head dimension of 64. Plus (8) indicates that LiSA$_{plus}$ is applied to 8 specific layers. The layer-wise sharing configuration for DS (8) and Plus (8) is ''2, 3, 4, 6, 7, 8, 10, 11''. For the GQA model, we set the number of KV attention heads to 2.

responses was better. Aligning with Wang et al. (2024), during the fine-tuning stage, we set the global batch size to 128, weight decay to 0, number of training epochs to 3, warmup steps to 0, maximal text length to 1,024, and the learning rate to 0.0001. In the generation stage, the decoding temperature was set to 0.75 and Top-p was set to 0.95 to ensure the diversity of generated responses.

C.3 Experimental Results

The experimental results presented in this section can be categorized as follows (aligning with the order in which they appear in the main text):

Visualization of the Attention Heads Alignment Module. Figure 14 visualizes the weight of a well-trained single-layer FFN for aligning attention heads. Values in two square matrices represent the learned weights accounting for the difference matrix A_Δ and the shared attention weight matrix A_{n-1}, respectively. Cells with either deeper blue or deeper red indicate the larger absolute value of weights. We can see that FFN assigns significant weights to the A_{n-1}, which carries most of the attention information. Moreover, there are also some deep blue cells in the left matrix, demonstrating the necessity of A_Δ.

Normal Forward. Table 10 compares the performance of enabling LiSA at all inference steps

versus activating LiSA only after the first inference step, during which standard forward computation is used.

Pre-train from Scratch. Figure 15 and Table 5 detail the training process and final performance of different attention models pre-trained from scratch.

Task-specific LiSA. Table 9 summarizes the performance of DS and LiSA, as well as their respective degradation compared to the original model. Based on the performance drop introduced by DS, tasks can be categorized to guide the configuration of LiSA, enabling a better balance between inference efficiency and model effectiveness.

Layer	1			1024			PIQA			GSM8K		
	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token
0	0.43	2	a	0.05	1025	her	0.17	36	.\n	0.09	699	?\n
0	0.19	3	time	0.05	1024	in	0.09	15	?\n	0.07	701	:
0	0.14	4	there	0.03	1019	.	0.08	5	\n	0.05	700	A
0	0.13	0	Once	0.03	58	,	0.08	26	,	0.02	688	.
0	0.11	1	upon	0.02	1023	was	0.07	22	,	0.02	694	of
1	0.62	2	a	0.61	58	,	0.51	22	,	0.55	27	,
1	0.19	1	upon	0.17	9	.	0.23	15	?\n	0.17	4	
1	0.10	3	time	0.04	1025	her	0.03	36	.\n	0.05	700	A
1	0.06	4	there	0.02	1024	in	0.02	5	\n	0.04	699	?\n
1	0.04	0	Once	0.02	1026	room	0.02	4	:	0.02	701	:
2	0.93	0	Once	0.79	0	Once	0.81	0	The	0.78	0	Q
2	0.03	3	time	0.01	1024	in	0.02	16	A	0.02	699	?\n
2	0.02	1	upon	0.01	1011	and	0.02	15	?\n	0.01	655	Axel
2	0.01	2	a	0.01	1019	.	0.02	17	:	0.01	654	:
2	0.01	4	there	0.01	1023	was	0.01	23	drain	0.01	600	Olivia
3	0.89	0	Once	0.84	0	Once	0.76	0	The	0.80	0	Q
3	0.06	3	time	0.02	1025	her	0.02	36	.\n	0.02	701	:
3	0.03	1	upon	0.02	1024	in	0.02	17	:	0.02	699	?\n
3	0.02	4	there	0.01	1023	was	0.02	5	\n	0.02	700	A
3	0.01	2	a	0.01	1026	room	0.02	15	?\n	0.01	654	:
4	0.90	0	Once	0.75	0	Once	0.66	0	The	0.72	0	Q
4	0.05	3	time	0.03	1025	her	0.04	15	?\n	0.03	701	:
4	0.03	4	there	0.02	1026	room	0.03	36	.\n	0.03	699	?\n
4	0.02	1	upon	0.02	1024	in	0.03	35	cream	0.02	700	A
4	0.01	2	a	0.02	1022	girl	0.02	33	and	0.01	654	:
5	0.90	0	Once	0.71	0	Once	0.62	0	The	0.70	0	Q
5	0.04	1	upon	0.02	1024	in	0.05	15	?\n	0.03	699	?\n
5	0.02	3	time	0.02	1025	her	0.04	17	:	0.02	701	:
5	0.02	4	there	0.02	1019	.	0.03	36	.\n	0.02	654	:
5	0.01	2	a	0.02	1023	was	0.02	33	and	0.01	689	What
6	0.88	0	Once	0.56	0	Once	0.53	0	The	0.62	0	Q
6	0.05	3	time	0.06	1025	her	0.05	36	.\n	0.04	701	:
6	0.03	1	upon	0.03	1019	.	0.04	15	?\n	0.04	699	?\n
6	0.03	4	there	0.03	1023	was	0.03	17	:	0.02	700	A
6	0.02	2	a	0.03	1024	in	0.03	35	cream	0.01	688	.
7	0.87	0	Once	0.66	0	Once	0.48	0	The	0.57	0	Q
7	0.04	1	upon	0.03	1025	her	0.08	15	?\n	0.05	701	:
7	0.04	4	there	0.02	1019	.	0.04	36	.\n	0.04	699	?\n
7	0.03	3	time	0.02	1023	was	0.03	17	:	0.03	654	:
7	0.02	2	a	0.02	1024	in	0.03	35	cream	0.02	688	.
8	0.83	0	Once	0.61	0	Once	0.46	0	The	0.30	0	Q
8	0.07	1	upon	0.02	1026	room	0.07	36	.\n	0.06	701	:
8	0.04	4	there	0.02	1025	her	0.06	15	?\n	0.06	699	?\n
8	0.03	3	time	0.01	1019	.	0.03	14	cheese	0.03	688	.
8	0.03	2	a	0.01	1023	was	0.03	17	:	0.03	689	What

Table 11: *Continued.*

Layer	1			1024			PIQA			GSM8K		
	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token
9	0.87	0	Once	0.52	0	Once	0.41	0	The	0.30	0	Q
9	0.05	1	upon	0.02	1025	her	0.06	36	.\n	0.06	699	?\n
9	0.03	3	time	0.02	1024	in	0.05	15	?\n	0.05	701	:
9	0.03	2	a	0.02	1026	room	0.03	35	cream	0.04	688	.
9	0.02	4	there	0.02	1023	was	0.03	4	:	0.03	689	What
10	0.89	0	Once	0.59	0	Once	0.55	0	The	0.31	0	Q
10	0.05	1	upon	0.03	1026	room	0.05	36	.\n	0.05	701	:
10	0.03	2	a	0.01	1024	in	0.04	15	?\n	0.04	699	?\n
10	0.02	4	there	0.01	1025	her	0.03	4	:	0.03	688	.
10	0.02	3	time	0.01	1019	.	0.03	35	cream	0.02	700	A
11	0.88	0	Once	0.62	0	Once	0.49	0	The	0.30	0	Q
11	0.06	1	upon	0.02	1019	.	0.06	15	?\n	0.06	701	:
11	0.03	3	time	0.02	1024	in	0.04	36	.\n	0.05	699	?\n
11	0.02	2	a	0.01	1023	was	0.04	16	A	0.03	688	.
11	0.01	4	there	0.01	1025	her	0.03	14	cheese	0.03	666	.
12	0.83	0	Once	0.64	0	Once	0.30	0	The	0.36	0	Q
12	0.07	1	upon	0.05	1026	room	0.08	36	.\n	0.11	701	:
12	0.05	3	time	0.03	1024	in	0.06	17	:	0.04	699	?\n
12	0.03	4	there	0.03	1019	.	0.06	15	?\n	0.03	688	.
12	0.02	2	a	0.03	1025	her	0.05	16	A	0.03	666	.
13	0.87	0	Once	0.51	0	Once	0.40	0	The	0.17	0	Q
13	0.06	1	upon	0.02	1026	room	0.07	16	A	0.08	666	.
13	0.03	4	there	0.01	1019	.	0.06	15	?\n	0.06	701	:
13	0.02	2	a	0.01	1024	in	0.04	36	.\n	0.05	688	.
13	0.02	3	time	0.01	1020	The	0.04	4	:	0.04	682	and
14	0.85	0	Once	0.43	0	Once	0.37	0	The	0.24	0	Q
14	0.06	3	time	0.04	1025	her	0.07	15	?\n	0.08	701	:
14	0.04	1	upon	0.02	1019	.	0.07	16	A	0.07	666	.
14	0.03	4	there	0.02	1024	in	0.05	36	.\n	0.04	699	?\n
14	0.02	2	a	0.02	1026	room	0.05	17	:	0.04	656	has
15	0.75	0	Once	0.40	0	Once	0.37	0	The	0.32	0	Q
15	0.10	3	time	0.04	1026	room	0.11	16	A	0.08	701	:
15	0.07	1	upon	0.03	1024	in	0.06	36	.\n	0.04	699	?\n
15	0.05	4	there	0.02	1025	her	0.04	7	:	0.03	700	A
15	0.03	2	a	0.02	1023	was	0.04	17	:	0.02	655	Axel
16	0.74	0	Once	0.55	0	Once	0.44	0	The	0.40	0	Q
16	0.11	3	time	0.04	1026	room	0.10	16	A	0.09	701	:
16	0.07	4	there	0.03	1025	her	0.07	36	.\n	0.05	655	Axel
16	0.06	1	upon	0.03	1019	.	0.04	15	?\n	0.05	700	A
16	0.03	2	a	0.02	1024	in	0.03	17	:	0.03	699	?\n
17	0.80	0	Once	0.55	0	Once	0.54	0	The	0.43	0	Q
17	0.11	3	time	0.04	1024	in	0.07	16	A	0.08	701	:
17	0.04	4	there	0.03	1025	her	0.07	36	.\n	0.03	700	A
17	0.03	1	upon	0.02	1026	room	0.03	15	?\n	0.02	699	?\n
17	0.01	2	a	0.02	1023	was	0.02	17	:	0.01	666	.

Table 11: *Continued.*

Layer	1			1024			PIQA			GSM8K		
	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token
18	0.86	0	Once	0.66	0	Once	0.65	0	The	0.55	0	Q
18	0.05	3	time	0.02	1024	in	0.07	16	A	0.06	701	:
18	0.04	1	upon	0.02	1023	was	0.04	36	.\n	0.03	655	Axel
18	0.03	4	there	0.02	1025	her	0.03	17	:	0.03	680	he
18	0.01	2	a	0.02	1022	girl	0.02	15	?\n	0.03	696	they
19	0.90	0	Once	0.69	0	Once	0.62	0	The	0.59	0	Q
19	0.04	4	there	0.02	1026	room	0.07	16	A	0.04	701	:
19	0.04	3	time	0.01	1024	in	0.06	7	:	0.03	655	Axel
19	0.02	1	upon	0.01	1025	her	0.05	36	.\n	0.02	699	?\n
19	0.01	2	a	0.01	1019	.	0.02	15	?\n	0.01	700	A
20	0.89	0	Once	0.61	0	Once	0.61	0	The	0.59	0	Q
20	0.05	3	time	0.02	1024	in	0.07	16	A	0.06	655	Axel
20	0.03	4	there	0.01	1026	room	0.06	36	.\n	0.04	701	:
20	0.02	1	upon	0.01	1023	was	0.05	7	:	0.01	700	A
20	0.01	2	a	0.01	1019	.	0.03	15	?\n	0.01	699	?\n
21	0.87	0	Once	0.68	0	Once	0.66	0	The	0.63	0	Q
21	0.06	3	time	0.04	1025	her	0.08	36	.\n	0.08	701	:
21	0.05	4	there	0.03	1026	room	0.05	16	A	0.02	698	together
21	0.01	1	upon	0.02	1024	in	0.04	15	?\n	0.02	699	?\n
21	0.01	2	a	0.01	1019	.	0.02	35	cream	0.02	700	A
22	0.89	0	Once	0.67	0	Once	0.66	0	The	0.63	0	Q
22	0.04	3	time	0.02	1026	room	0.08	36	.\n	0.05	701	:
22	0.04	4	there	0.02	1025	her	0.07	16	A	0.03	655	Axel
22	0.03	1	upon	0.01	1024	in	0.02	15	?\n	0.02	699	?\n
22	0.01	2	a	0.01	1019	.	0.02	5	\n	0.01	177	Originally
23	0.93	0	Once	0.77	0	Once	0.75	0	The	0.71	0	Q
23	0.04	3	time	0.02	1024	in	0.06	16	A	0.05	655	Axel
23	0.01	4	there	0.02	1026	room	0.05	36	.\n	0.03	701	:
23	0.01	1	upon	0.01	1025	her	0.02	15	?\n	0.01	700	A
23	0.00	2	a	0.01	1023	was	0.01	7	:	0.01	660	pesos
24	0.94	0	Once	0.72	0	Once	0.76	0	The	0.73	0	Q
24	0.03	4	there	0.02	1026	room	0.05	16	A	0.08	655	Axel
24	0.02	3	time	0.02	1019	.	0.05	36	.\n	0.03	701	:
24	0.01	1	upon	0.01	1025	her	0.02	7	:	0.01	660	pesos
24	0.01	2	a	0.01	1024	in	0.02	5	\n	0.01	699	?\n
25	0.92	0	Once	0.78	0	Once	0.82	0	The	0.79	0	Q
25	0.05	3	time	0.04	1025	her	0.05	36	.\n	0.04	701	:
25	0.02	4	there	0.02	1026	room	0.04	16	A	0.04	655	Axel
25	0.01	1	upon	0.01	1024	in	0.02	15	?\n	0.01	699	?\n
25	0.00	2	a	0.01	1019	.	0.01	5	\n	0.01	700	A
26	0.90	0	Once	0.63	0	Once	0.67	0	The	0.59	0	Q
26	0.04	3	time	0.02	1026	room	0.12	36	.\n	0.10	655	Axel
26	0.04	4	there	0.01	1019	.	0.05	16	A	0.07	701	:
26	0.02	1	upon	0.01	1023	was	0.03	15	?\n	0.02	700	A
26	0.01	2	a	0.01	1025	her	0.02	5	\n	0.02	699	?\n

Table 11: *Continued.*

Layer	1			1024			PIQA			GSM8K		
	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token	Weight	Index	Token
27	0.91	0	Once	0.74	0	Once	0.80	0	The	0.64	0	Q
27	0.03	4	there	0.02	1026	room	0.05	36	.\n	0.08	655	Axel
27	0.03	3	time	0.01	1025	her	0.03	16	A	0.04	701	:
27	0.02	1	upon	0.01	724	the	0.02	35	cream	0.01	700	A
27	0.01	2	a	0.01	1019	.	0.01	15	?\n	0.01	699	?\n
28	0.84	0	Once	0.64	0	Once	0.68	0	The	0.64	0	Q
28	0.07	4	there	0.05	1026	room	0.11	36	.\n	0.10	701	:
28	0.06	3	time	0.03	1025	her	0.03	16	A	0.03	700	A
28	0.02	1	upon	0.01	1024	in	0.03	15	?\n	0.03	655	Axel
28	0.01	2	a	0.00	1023	was	0.02	35	cream	0.02	699	?\n
29	0.88	0	Once	0.67	0	Once	0.75	0	The	0.76	0	Q
29	0.06	4	there	0.04	1026	room	0.06	36	.\n	0.05	701	:
29	0.05	3	time	0.02	1025	her	0.04	16	A	0.01	700	A
29	0.01	1	upon	0.01	1019	.	0.03	15	?\n	0.01	699	?\n
29	0.01	2	a	0.00	1018	room	0.01	5	\n	0.01	695	pesos
30	0.85	0	Once	0.49	0	Once	0.60	0	The	0.52	0	Q
30	0.08	4	there	0.05	1026	room	0.11	36	.\n	0.09	701	:
30	0.04	3	time	0.02	1019	.	0.04	16	A	0.04	655	Axel
30	0.03	1	upon	0.01	1025	her	0.03	15	?\n	0.02	700	A
30	0.01	2	a	0.01	1024	in	0.02	6	Q	0.02	699	?\n
31	0.78	0	Once	0.49	0	Once	0.61	0	The	0.51	0	Q
31	0.16	4	there	0.16	1026	room	0.21	36	.\n	0.22	701	:
31	0.04	3	time	0.02	1025	her	0.02	15	?\n	0.02	699	?\n
31	0.01	1	upon	0.02	1019	.	0.02	16	A	0.01	700	A
31	0.01	2	a	0.01	1024	in	0.01	17	:	0.01	655	Axe

Table 11: Top-5 tokens with the highest attention weights in each layer of LLaMA3-8B. ‘‘1’’ and ‘‘1024’’ refer to snapshots taken at generation steps 1 and 1024, respectively. ‘‘PIQA’’ and ‘‘GSM8K’’ show the results obtained by inputting a randomly selected sample from the PIQA and GSM8K benchmarks.