EchoAtt: Attend, Copy, then Adjust for More Efficient Large Language Models

Hossein Rajabzadeh[*,1,2], **Aref Jafari**[*,1,2], **Aman Sharma**[1,2], **Benyamin Jami**[2], **Hyock Ju Kwon**[1], **Ali Ghodsi**[1], **Boxing Chen**[2], **Mehdi Rezagholizadeh**[2]
[1] University of Waterloo, [2] Huawei Noah's Ark Lab, [*] Equal contributions
{hossein.rajabzadeh, aref.jafari, aman.sharma, hjkwon, ali.ghodsi}@uwaterloo.ca
{mehdi.rezagholizadeh, benyamin.jami, boxing.chen}@huawei.com

Abstract

Large Language Models (LLMs), with their increasing depth and number of parameters, have demonstrated outstanding performance across a variety of natural language processing tasks. However, this growth in scale leads to increased computational demands, particularly during inference and fine-tuning. To address these challenges, we introduce **EchoAtt**, a novel framework aimed at optimizing transformer-based models by analyzing and leveraging the similarity of attention patterns across layers. Our analysis reveals that many inner layers in LLMs, especially larger ones, exhibit highly similar attention matrices. By exploiting this similarity, **EchoAtt** enables the sharing of attention matrices in less critical layers, significantly reducing computational requirements without compromising performance. We incorporate this approach within a knowledge distillation setup, where a pre-trained teacher model guides the training of a smaller student model. The student model selectively shares attention matrices in layers with high similarity while inheriting key parameters from the teacher. Our best results with TinyLLaMA-1.1B demonstrate that **EchoAtt** improves inference speed by 15%, training speed by 25%, and reduces the number of parameters by approximately 4%, all while improving zero-shot performance. These findings highlight the potential of attention matrix sharing to enhance the efficiency of LLMs, making them more practical for real-time and resource-limited applications.

1 Introduction

In recent years, Large Language Models (LLMs) have made significant strides in natural language processing (NLP) and extended their reach across a variety of fieldsYang et al. [2024], Wei et al. [2022], Patil et al. [2023], Tahaei et al. [2024], revolutionizing applications such as machine translation, text generation, and question answering. The success of these models can largely be attributed to the transformer architecture Vaswani [2017], which employs a self-attention mechanism that enables the model to capture contextual relationships between words more effectively than traditional models. However, as the size of these models grows, the computational complexity and memory requirements scale significantly, with a complexity of $O(n^2)$ for self-attention and $O(n)$ for memory footprint. This growing computational demand creates a bottleneck, particularly during inference and fine-tuning, making these models challenging to deploy in real-time or resource-constrained environments.

Numerous strategies have been proposed to mitigate the computational inefficiency of transformers Zhang et al. [2024a], Rajabzadeh et al. [2024], Lieber et al. [2024], Beck et al. [2024], including the development of alternative architectures like linear attention models Arora et al. [2024], Yang et al. [2023], Gu and Dao [2023]. However, these models often struggle to match the generalization

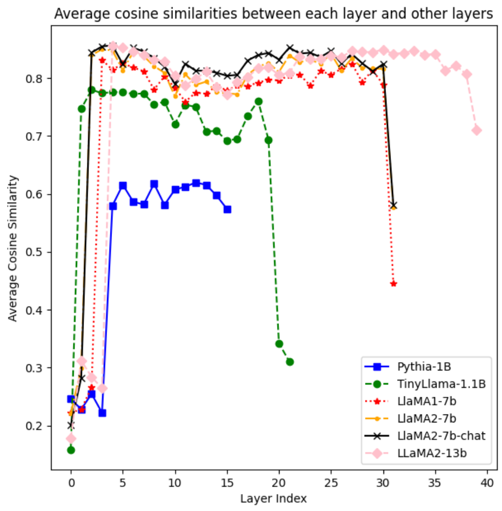


Figure 1: Average cosine similarities between one layer's attention and other layers' attentions. The results demonstrate that attention scores in some layers are more similar than that of the other layers.

and performance capabilities of standard transformer models. Addressing this trade-off between efficiency and performance remains a critical challenge.

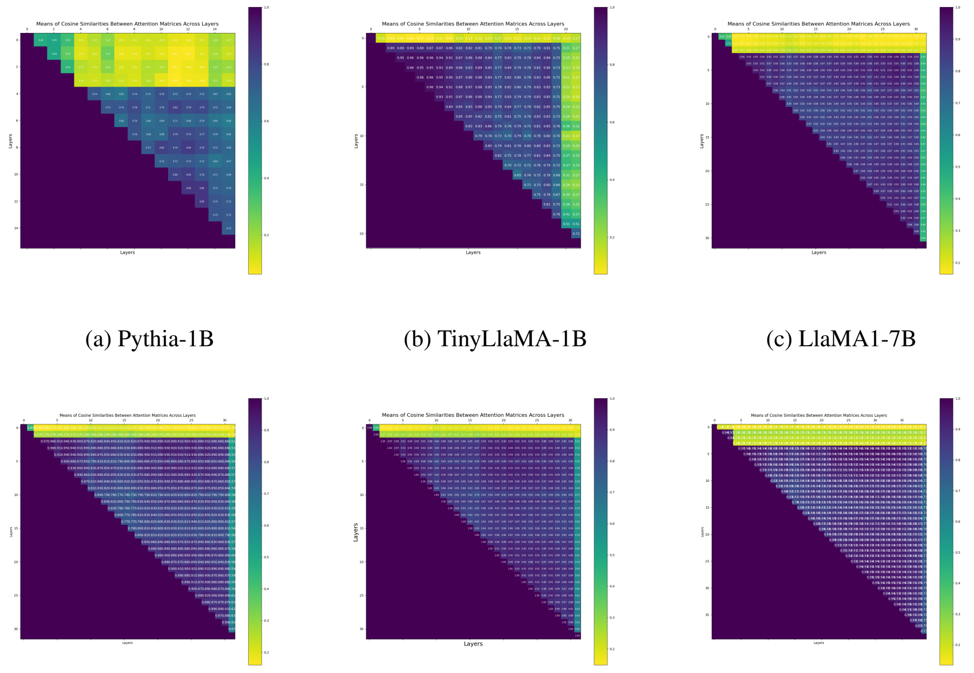
In this study, we propose a novel framework to address the inefficiencies of transformer-based LLMs while maintaining their performance. Through an in-depth analysis of attention patterns across different layers of transformers, we observe that in larger models, inner layers tend to exhibit highly similar attention matrices, particularly in generative models. This similarity becomes more pronounced with larger models, aligning with previous findings Ying et al. [2021], Bhojanapalli et al. [2021], He et al. [2024], Liao and Vargas [2024]. Leveraging this insight, we introduce a knowledge distillation-based framework Hinton [2015], Jafari et al. [2021], which selectively shares attention mechanisms between layers exhibiting high similarity. Our method reduces the number of parameters and computational costs by sharing attention patterns in less critical layers, while retaining unique attention mechanisms in the more distinct layers, typically located in the first and last layers of the network.

To validate our approach, we apply it to the TinyLLaMA-1.1B model and conduct extensive experiments to assess the impact on performance and efficiency. Our results show that by sharing inner attention matrices, we can reduce the parameter count by 3.86%, while improving inference speed by 15% and training speed by 25%. Moreover, this compression comes with minimal loss in accuracy, maintaining competitive performance in zero-shot settings across various benchmarks.

This study not only offers a method for reducing the computational complexity of LLMs but also provides insights into how selective sharing of attention patterns can optimize both the performance and resource efficiency of these models. The contributions of this paper can be summarized as: 1) introducing **EchoAtt**, a novel framework designed to optimize transformer-based Large Language Models (LLMs) by leveraging the similarity of attention patterns across layers, 2) proposing a method for **attention matrix sharing** in less critical layers, significantly reducing computational requirements while maintaining model performance, 3) integrating this approach within a **knowledge distillation** setup, and 4) demonstrating that **EchoAtt** improves inference and training speed and also reduces the number of parameters, while maintaining competitive zero-shot performance.

2 Analysis

To analyze the similarity between attention scores across different layers, we employ a subset of the IMDB dataset Maas et al. [2011]. In this subset, each sample is standardized to a length of 512 tokens, effectively eliminating the need for padding and normalizing the sequence length across



(a) Pythia-1B (b) TinyLlaMA-1B (c) LlaMA1-7B

(d) LlaMA2-7B (e) LlaMA2-7B Chat (f) LlaMA2-13B

Figure 2: Average cosine similarities between the attention matrices of different layers in various LLMs, visualized as upper-triangle matrices. Each entry $[i, j]$ represents the similarity between the attention scores of layer *i* and layer *j*, with higher values indicating more similar attention mechanisms. The results highlight attention similarities in inner layers, suggesting potential for sharing attention mechanisms to reduce computational complexity.

batches. For each sample, we perform a forward pass and record the attention scores at each layer. We then use cosine similarity as a metric to measure the similarities between the flattened attention scores across the head and embedding dimensions. The results are obtained for several LLMs and illustrated in Figure 2, where each sub-figure depicts the average cosine similarities between attention scores of every pair of layers. For instance, the entry $[i, j]$ in Sub-figure 2b and 2f, respectively, depict the average cosine similarities between attention scores of layer i and layer j in TinyLlaMA-1B Zhang et al. [2024b] and LlaMA2-13B Touvron et al. [2023] [1]. Moreover, Figure 1 demonstrates the average of attention scores between each layer and all other layers. The results indicate that a significant number of layers share similar attention scores, while a few layers, typically the first and last few layers, exhibit distinct attention patterns. This analysis offers a method for identifying which layers produce the most unique attention scores and which layers can potentially share attention mechanisms. For example, in LLaMA2-13B, the first four layers and the last layer demonstrate the most distinctive attention scores, whereas the other layers display more similar attention patterns.

By identifying layers with highly similar attention scores, we can explore strategies to reduce model complexity, such as attention mechanism sharing. Conversely, recognizing layers with unique attention patterns can guide targeted improvements in model design, ensuring that these critical components are preserved and enhanced. However, naively sharing similar attention scores across layers can lead to performance degradation in shared attention, especially as the number of shared layers increases. To address this issue, the following sections introduce a knowledge distillation mechanism combined with continual training [2], which significantly enhances the performance of shared attention.

[1]Results are reported in an upper-triangle matrix, as the cosine similarity between attention scores is symmetric, i.e. $cosine(Attn_{l_i}, Attn_{l_j}) = cosine(Attn_{l_j}, Attn_{l_i})$

[2]Continual training refers to starting training from a pre-trained checkpoint.

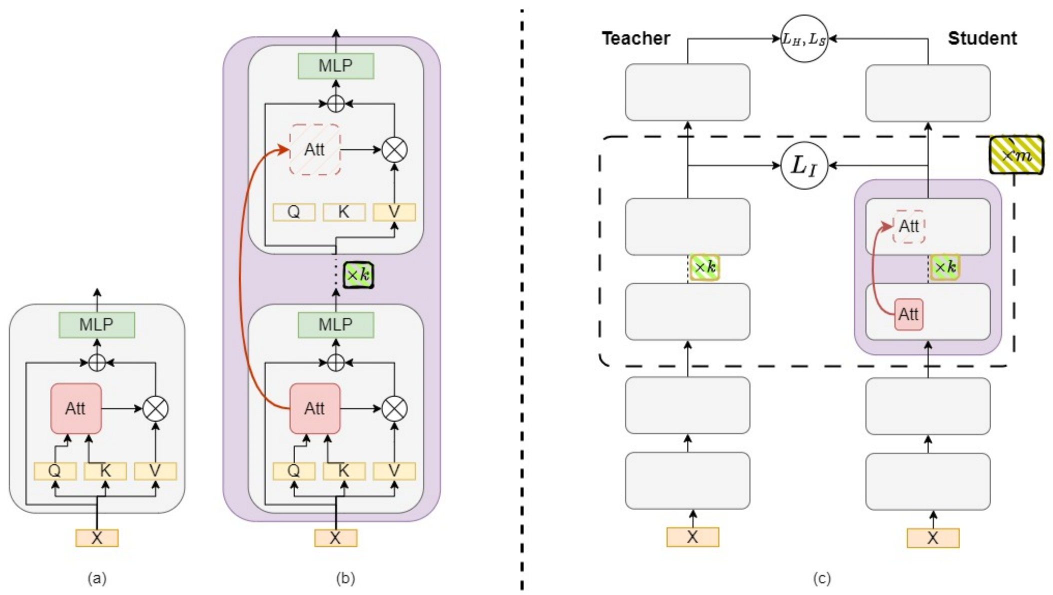


Figure 3: (a) A standard transformer block, which consists of a single transformer layer. (b) A shared attention block, where multiple transformer layers utilize a single attention mechanism. (c) The architecture of the student and teacher models used in the proposed distillation method.

3 Method: Shared Attention

The goal of the proposed method is to reduce the computational complexity and memory footprint of transformer-based LLMs by identifying and sharing similar attention patterns across layers. We build upon the observation that many inner layers of these models exhibit highly similar attention matrices, and we propose a framework that leverages this similarity to share attention mechanisms across layers, reducing the number of parameters and computational cost. The method is divided into two main stages: constructing a shared attention student model and applying knowledge distillation from a pre-trained teacher model.

3.1 Model Construction

The primary observation driving this work is the high similarity between the attention matrices in the inner layers of transformer models. Specifically, we compute cosine similarity between the attention matrices of different layers and find that many inner layers produce nearly identical attention patterns. Based on this analysis, we design a **shared attention model** similar to the approach described by Ying et al. [2021] that optimizes computational efficiency by sharing attention matrices across layers with high similarity.

To construct the student model, we retain the first and last few layers unchanged. To determine which layers to retain, we compute the average cosine similarity of each layer with all other layers, as depicted in Figure 1. The layers are then sorted based on their similarity scores, and the cutoff point is set to the maximum distance between similarity scores. Layers with smaller similarity scores are considered unchanged. Additionally, we impose a constraint that the unchanged layers must be among the first or last layers. For the inner layers, we implement a shared attention mechanism similar to the approach described by Ying et al. [2021]. Specifically, every k consecutive inner layers are grouped into what we refer to as a *shared attention block* (see Figure 3-b), where attention matrices are shared among these blocks. This design significantly reduces computational time and memory footprint of the student model by avoiding the computation of separate attention matrices for the shared layers and eliminating the K and Q values in these layers, with minimal impact on performance. The hyperparameter k controls the extent of model compression achieved through this design.

3.1.1 Shared Attention Blocks

In standard transformers, each layer computes its own attention matrix using the query, key, and value matrices Q_i, K_i, and V_i. In the proposed shared attention model (see Figure 3b), for each shared attention block, a single set of Q and K matrices is computed and used for all layers within the block. This reduces the number of parameters and avoids redundant computations of highly similar attention matrices.

Formally, the attention mechanism in standard transformers is defined as:

$$\text{Att}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d}}\right) V_i$$

where i is the layer index.

In our shared attention model, for every k consecutive layers with similar attention matrices, we compute a shared attention mechanism as:

$$\text{A}_{shared} = \text{softmax}\left(\frac{Q_{shared} K_{shared}^T}{\sqrt{d}}\right)$$

$$\text{Att}_j = A_{shared} V_j, \qquad j \in [i, i+k]$$

where i is the shared attention block index.

3.1.2 Parameter Sharing Strategy

The parameter-sharing mechanism is controlled by a hyperparameter k, which dictates the number of consecutive layers that share attention matrices. A larger value of k results in more aggressive parameter sharing and greater model compression, while smaller values of k preserve more unique attention patterns.

3.2 Knowledge Distillation

Once the shared attention student model is constructed, we employ a **knowledge distillation** approach to transfer knowledge from a pre-trained teacher model to the student model. This process helps recover performance lost due to the parameter sharing and ensures that the student model achieves competitive results.

3.2.1 Distillation Setup

The knowledge distillation process consists of two stages:

- **Stage 1: Distillation with teacher's Pseudo-Labels** In the first stage, the student model is trained using the outputs of the teacher model as pseudo-labels. Both the student and teacher models are fed the same input tokens, and the student is trained to match the teacher's output at multiple levels. Three loss functions are used to guide the distillation process (see Figure 3c):
  - **Intermediate Layer Loss ($\mathcal{L}_I$)**: This loss aligns the intermediate layer outputs of the student and teacher models for each shared attention block. We use mean squared error to minimize the distance between the shared layers of the student and the corresponding layers of the teacher.

    $$\mathcal{L}_I = \frac{1}{m} \sum_{i=1}^{m} \| S_{ki+b}(x) - T_{ki+b}(x) \|_2^2$$

    where m represents the number of shared attention blocks, k denotes the number of attention layers within each shared block, and b indicates the number of early layers that are skipped. S_j and T_j refer to the outputs of the student and teacher models at layer j.

Table 1: Performance comparison of the Shared-Attention TinyLlaMA model across different attention-sharing ratios in a zero-shot evaluation. The table presents accuracy metrics obtained under continual training conditions **without** any knowledge distillation. Baseline results are compared against three variations of the model with 77%, 41%, and 23% attention-sharing ratios.

Benchmark	TinyLlaMA(baseline)	Ours (77%)	Ours (41%)	Ours (23%)
mmlu	25.27 ± 0.37	23.44 ± 0.36	25.05 ± 0.37	25.43 ± 0.50
winogrande	59.35 ± 1.38	50.75 ± 0.14	54.54 ± 0.14	58.41 ± 1.39
swag	51.69 ± 0.35	46.29 ± 0.35	49.80 ± 0.35	51.0 ± 0.35
hellaswag	46.40 ± 0.05	36.57 ± 0.49	43.35 ± 0.49	44.50 ± 0.50
xnli_en	41.53 ± 0.99	45.22 ± 0.01	52.64 ± 0.01	52.61 ± 0.01
agieval_en	17.03 ± 0.01	16.77 ± 0.01	17.39 ± 0.01	17.19 ± 0.01
Average	40.21	36.50	39.51	**41.52**

- **Soft Label Loss** ($\mathcal{L}_S$): A KL-divergence loss is used to match the soft label distributions of the student and teacher models. This loss ensures that the student learns from the probability distributions produced by the teacher model.

$$\mathcal{L}_S = \mathrm{KL}(\sigma(S(x))\|\sigma(T(x)))$$

- **Hard Label Loss** ($\mathcal{L}_H$): Cross-entropy loss is applied to distill hard labels sampled from the teacher model into the student model. This step helps the student model learn from the teacher's confident predictions.

$$\mathcal{L}_H = \mathrm{CE}(\sigma(S(x)), \tau(T(x)))$$

The functions σ and τ denote the softmax and argmax functions, respectively. and $S(x)$ and $T(x)$ are student and teacher models outputs.

The final loss function is a weighted combination of these three components:

$$\mathcal{L} = \alpha\mathcal{L}_I + \beta\mathcal{L}_S + \gamma\mathcal{L}_H$$

where α, β, and γ are tunable coefficients controlling the contribution of each loss function.

- **Stage 2: Refinement with True Labels** In the second stage, the student model is further fine-tuned using the actual labels from the training dataset. This stage allows the student to refine its predictions and improve its accuracy. Cross-entropy loss is used for this step, and the student is trained to directly predict the true labels from the input data.

4 Evaluations and Results

4.1 Experimental Setup

For all training experiments, we employed a random subset of the Slim-Pajama dataset Soboleva et al. [2023], comprising over 3.7 billion tokens. During the knowledge distillation and continual training stages, the models were trained for 1 epoch and 0.25 epochs, respectively, on this dataset. A detailed list of the critical hyper-parameters used in our experiments is provided in Table 6 in Appendix A. It is important to note that no hyper-parameter fine-tuning was applied during the experiments. The hyper-parameters were kept consistent across all stages of training to ensure that the results reflect the true performance of the models under identical conditions, without any optimization specific to individual tasks or datasets. Additionally, we used LLaMA-Factory for training Zheng et al. [2024] and LM-Evaluation-Harness Gao et al. [2024] for evaluation.

4.2 Results

To validate the efficacy of our model, we employ TinyLlaMA Zhang et al. [2024b][3] as our baseline LLM. Tables 1 and 2 compares the accuracy of the baseline against its shared attention versions where a certain percentage of attention layers are shared across the network, indicated by the sharing ratios 77%, 41%, and 23%. Table 1 demonstrates the shared attention performance with just

[3]`https://huggingface.co/TinyLlama/TinyLlama_v1.1`

Table 2: Performance comparison of the Shared-Attention TinyLlaMA model across different attention-sharing ratios in a zero-shot evaluation. The table presents accuracy metrics obtained under continual training conditions, **coupled with** knowledge distillation. Baseline results are compared against three variations of the model with 77%, 41%, and 23% attention-sharing ratios, demonstrating the impact of varying the degree of attention sharing on overall model performance.

Benchmark	TinyLlaMA (baseline)	Ours (77%)	Ours (41%)	Ours(23%)
mmlu	25.27 ± 0.37	24.55 ± 0.36	25.96 ± 0.37	25.94 ± 0.50
winogrande	59.35 ± 1.38	52.33 ± 1.40	58.01 ± 1.39	58.36 ± 1.4
swag	51.69 ± 0.35	47.96 ± 0.35	50.33 ± 0.35	50.78 ± 0.35
hellaswag	46.40 ± 0.05	38.56 ± 0.49	43.82 ± 0.49	44.40 ± 0.50
xnli_en	41.53 ± 0.99	49.32 ± 1.0	53.25 ± 1.0	52.97 ± 0.01
agieval_en	17.03 ± 0.01	17.50 ± 0.59	17.94 ± 0.59	17.21 ± 0.01
Average	40.21	38.37	**41.55**	**41.61**

Table 3: Comparing the baseline TinyLlaMA-1.1B against its shared attention versions in terms of speedup in training, inference, and the reduced portion of parameters. The inference speed is reported based on one 32GiG-V100 GPU, and the training speed is computed by eight 46GiG-L40-GPUs.

Model	Inference Speed	Training Speed	Reduced Parameters
TinyLlaMA (baseline)	28.44 token/sec	43h,30m	None
Shared TinyLlaMA(23%)	30.99 (9%faster)	37h,30m (14%faster)	24 millions (2.14%)
Shared TinyLlaMA(41%)	32.61 (15%faster)	32h,50m (25% faster)	43 millions (3.86%)
Shared TinyLlaMA(77%)	40.50 (42%faster)	23h,30m (46%faster)	80 millions (7.29%)

continual training while Table 2 repeats the same experiments with both knowledge distillation and continual training. The results show that with continual training only, the model with a 23% sharing ratio outperforms the baseline, the 41% ratio performs comparably to the baseline, and the 77% ratio underperforms. However, when shared attention is combined with both knowledge distillation and continual training, the models with 23% and 41% sharing ratios outperform the baseline, while the performance gap for the model with 77% sharing is significantly reduced. The superior performance of the models with 23% and 41% sharing ratios could be attributed to a slight regularization effect of shared attention, as noted by Bondarenko et al. [2024]. In this context, sharing attention leads to a reduction in the number of parameters, which may act as a form of regularization, thereby enhancing model performance.

Overall, the results indicate that knowledge distillation coupled with continual training improves the performance of shared attention in all sharing ratios.

Table 3 compares the baseline TinyLLaMA-1.1B model with its shared attention versions, focusing on inference speed, training speed, and the reduction in the number of parameters. The shared attention models demonstrate notable improvements in both inference and training speeds. Specifically, the model with 77% shared attention achieves the highest performance gains, with a 42% increase in inference speed, reaching 40.50 tokens per second, compared to the baseline's 28.44 tokens per second. In terms of training efficiency, this same model reduces the training time by 46%, completing the process in 23 hours and 30 minutes, down from the baseline's 43 hours and 30 minutes. Additionally, the shared attention models also exhibit a reduction in the total number of parameters. The 77% shared attention model reduces the number of parameters by $\approx$ 80 million, corresponding to a 7.29% reduction. The other shared models follow a similar trend, with the 23% and 41% shared attention models achieving reductions of 24 million (2.14%) and 43 million (3.86%) parameters, respectively. Overall, these results indicate that the proposed approach of shared attention not only accelerates both inference and training processes but also leads to a more parameter-efficient model, making it a promising technique for optimizing large language models.

Table 4: Zeros-shot evaluation of TinyLlaMA before and after continual training. No attention sharing is applied here. The results demonstrate a slight decrease in average performance, suggesting that continual training may not be beneficial for this model under the tested conditions.

Benchmark	TinyLlaMA-1.1B (baseline)	Continual trained TinyLlaMA
mmlu	25.27 ± 0.37	25.39 ± 0.37
winogrande	59.35 ± 1.38	58.41 ± 1.38
swag	51.69 ± 0.35	51.44 ± 0.32
hellaswag	46.40 ± 0.05	46.10 ± 0.05
agieval_en	17.03 ± 0.01	16.98 ± 0.59
TruthfulQA_mc1	22.28 ± 1.50	20.69 ± 1.40
TruthfulQA_mc2	35.09 ± 1.40	34.08 ± 1.40
Average	36.69	36.15

Table 5: Zero-shot evaluation of the Distilled-Shared LlaMA-160m in terms of accuracy before and after continual training. Shared attention ratio is set to 33%. The results indicate that continual training, when combined with knowledge distillation, outperforms the approach where continual training is omitted from the shared attention training process.

Benchmark	LlaMA-160m (baseline)	Distilled-Shared-Attn	Continual-Distilled-Shared-Attn
mmlu	23.02	22.97	22.97
winogrande	50.12	51.70	51.54
swag	40.21	36.39	38.91
hellaswag	30.94	29.32	29.92
agieval_en	17.52	16.95	17.86
Average	32.36	31.46	32.24

4.3 Ablation Study

This ablation study aims to determine whether continual training can enhance the performance of our baseline model. To investigate this, we subjected our baseline model, TinyLlaMA without attention sharing, to continual training using the same dataset and settings as the shared attention models. The results, presented in Table 4, indicate that continual training not only fails to improve TinyLlaMA's performance but actually leads to a slight decrease in its average performance. Therefore, continual training does not benefit the vanilla TinyLlaMA model and may even be detrimental under the conditions tested.

The next ablation study evaluates the impact of continual training on top of distillation stage based on the performance of shared attention models. To that end, we employ LlaMA-160m Miao et al. [2023] and, first, train the shared attention version of this model with a sharing ratio of 33% (the indices of shared layers are: [4,6,8,10]) while excluding the continual training stage. Let's call this model as Distilled-Shared-Attn. Then, we allow the Distilled-Shared-Attn to receive the continual training stage, and refer to this model by Continual-Distilled-Shared-Attn. Table 5 demonstrates the performance of these two models and compare them with the baseline, i.e. LlaMA-160m. The results clearly show that the shared attention models achieve competitive performance compared to the baseline. Notably, the performance of the shared attention models further improves when continual training is applied on top of knowledge distillation, demonstrating the effectiveness of this approach in enhancing model accuracy and generalization. This observation highlights the potential of shared attention mechanisms in reducing model complexity without compromising performance, especially when combined with advanced training techniques.

5 Conclusion

We investigated attention mechanisms in large language models and proposed a framework that identifies and shares less important attentions, coupled with knowledge distillation and continual training to recover performance. Our experiments demonstrated that, on TinyLLaMA-1.1B, this approach improved average zero-shot performance, increased training and inference speeds up to 42%

and 46%, respectively, and reduced parameters by 7.29%. These results highlight the effectiveness of selective attention sharing in enhancing model efficiency without compromising performance.

References

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Marzieh Tahaei, Aref Jafari, Ahmad Rashid, David Alfonso-Hermelo, Khalil Bibi, Yimeng Wu, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. Efficient citer: Tuning large language models for enhanced answer quality and verification. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4443–4450, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.277. URL `https://aclanthology.org/2024.findings-naacl.277`.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024a.

Hossein Rajabzadeh, Mojtaba Valipour, Tianshu Zhu, Marzieh Tahaei, Hyock Ju Kwon, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. Qdylora: Quantized dynamic low-rank adaptation for efficient large language model tuning. *arXiv preprint arXiv:2402.10462*, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

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

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

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*, 2024.

Bingli Liao and Danilo Vasconcellos Vargas. Beyond kv caching: Shared attention for efficient llms. *arXiv preprint arXiv:2407.12866*, 2024.

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. Annealing knowledge distillation. *arXiv preprint arXiv:2104.07163*, 2021.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 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, 2024. Association for Computational Linguistics. URL `http://arxiv.org/abs/2403.13372`.

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. A framework for few-shot language model evaluation, 07 2024. URL `https://zenodo.org/records/12608602`.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. Low-rank quantization-aware training for llms. *arXiv preprint arXiv:2406.06385*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification, 2023.

A Hyper-Parameters

The complete list of hyper-parameters used in our experiments is detailed in Table 6. It is important to note that no hyper-parameter fine-tuning was applied during the experiments. The hyper-parameters were kept consistent across all stages of training to ensure that the results reflect the true performance of the models under identical conditions, without any optimization specific to individual tasks or datasets.

Table 6: List of essential hyperparameters used in the experiments, detailing the key settings that governed model training.

Name	Value
Optimizer	AdamW
Deepspeed stage	Zero3
Learning rate (lr)	1e-4
Lr scheduling type	Cosine
Max sequence length	2048
Global batch size	1024
FP16	True
Warmup ratio	0.005
$[\alpha, \beta, \gamma]$	$[0.25, 0.25, 0.5]$
LlaMA-160m: Shared layer indices (33%)	[4,6,8,10] out of 12 layers
TinyLlaMA: Shared layer indices (23%)	[2,5,4,3,7] out of 22 layers
TinyLlaMA: Shared layer indices (41%)	[2,5,4,3,7,6,18,9] out of 22 layers
TinyLlaMA: Shared layer indices (77%)	[2,5,4,3,7,6,18,9,8,11,12,1,17,10,14,13,16] out of 22 layers

B Limitation

Despite the promising results, our study has certain limitations that need to be considered. First, due to computational constraints, we were unable to evaluate the performance of shared attention mechanisms on larger language models (LLMs) [4]. Extending our experiments to encompass models with greater parameter counts would provide deeper insights into the scalability and effectiveness of our approach in more complex architectures. Such an evaluation could reveal potential challenges or benefits that are not apparent in smaller models like TinyLLaMA-1.1B.

Second, we did not investigate how supervised fine-tuning (SFT) operates within the shared attention framework for downstream tasks. Exploring the interaction between SFT and shared attention models could offer valuable information on how these models perform when adapted to specific applications, such as question answering.

[4]However, our analysis suggests that larger LLMs tend to exhibit similar attention patterns across layers, which could indicate that shared attention mechanisms might be particularly effective in these models as well.