

UniAttn: Reducing Inference Costs via Softmax Unification for Post-Training LLMs

Yizhe Xiong^{1,2,†} Wei Huang^{3,†} Xin Ye^{4,†} Hui Chen^{2*} Zijia Lin^{1*} Haoran Lian⁵
Zhenpeng Su⁶ Jungong Han⁷ Guiguang Ding^{1,2*}

¹School of Software, Tsinghua University ²BNRist, Tsinghua University

³School of Computer Science, BUPT ⁴Kuaishou Technology ⁵Beihang University

⁶UCAS ⁷Department of Automation, Tsinghua University

{xiongyizhe2001, jichenhui2012}@gmail.com linzijia07@tsinghua.org.cn dinggg@tsinghua.edu.cn

Abstract

Post-training is essential for adapting Large Language Models (LLMs) to real-world applications. Deploying post-trained models faces significant challenges due to substantial memory overhead and noticeable inference latency. Existing work has identified significant redundancies in LLMs and proposed efficient architectures, namely intra-layer KV sharing and cross-layer KV sharing. However, these methods still result in high inference time overhead, remaining suboptimal for post-training pre-trained LLMs. In this paper, we identify that the Softmax operation is a primary bottleneck for LLM inference and discover that it is actually highly redundant during post-training. We propose Softmax **Unification in Attention (UniAttn)**, a novel post-training method that unifies Softmax activations across transformer blocks to reduce LLM inference costs. Additionally, UniAttn adopts a linear projection to refine the approximation induced by Softmax unification. Experiments show that UniAttn matches the performance of standard post-training while significantly reducing inference costs, outperforming existing efficient architectures during post-training.

1 Introduction

Post-training (Lambert et al., 2024) has become a critical step in developing advanced LLMs for both general-purpose tasks (OpenAI, 2023; Liu et al., 2023) and domain-specific tasks (Thirunavukarasu et al., 2023; Wu et al., 2023; Shao et al., 2024). It typically refers to performing supervised fine-tuning (SFT) or domain-specific continual pre-training on pre-trained LLMs using a text corpus different from the original pre-training dataset. Current post-training approaches for real-world applications involve fine-tuning popular base models (Dubey et al., 2024; Jiang et al., 2023; Rivière et al., 2024) with standard decoder-based architectures

that have scaled to trillions of parameters. However, these approaches face two widely recognized challenges in model deployment (Xiao et al., 2024; Wang et al., 2024a; He et al., 2024; Brandon et al., 2024; Chen et al., 2024): (1) the substantial memory overhead required for KV-cache storage, and (2) the computational latency, which leads to noticeable inference time.

To address those deployment challenges, recent research has introduced *architecture-time* KV-cache sharing designs during pre-training. Depending on the granularity of redundancies, there are two mainstream paradigms: *intra-layer* KV sharing and *cross-layer* KV sharing. Intra-layer KV sharing, such as MQA (Shazeer, 2019) and GQA (Ainslie et al., 2023), reduces redundancy across attention heads within the same layer by sharing keys and values, and is now widely adopted in modern LLMs. Beyond that, cross-layer KV sharing, such as CLA (Brandon et al., 2024) and YOCO (Sun et al., 2024), further shares keys and values across multiple layers to pursue additional memory reduction. However, cross-layer sharing is less established in mainstream deployed LLMs and is particularly difficult to *retrofit* in the post-training regime: directly applying cross-layer KV sharing during post-training can cause noticeable performance degradation and still translate poorly to wall-clock speedups at deployment (see Figure 1).

To obtain a more comprehensive understanding of inference costs and cross-layer behaviors in deployed LLMs, we empirically analyze major architectural modules inside them. First, we systematically profile the FLOPs contributions and TTFT latency proportions for MHA and FFN. Table 1 shows that the *Softmax operation* in MHA significantly contributes to inference costs, although accounting for less than 1% of total FLOPs. Similar observations have also been reported in other Transformer-based architectures (Koohpayegani and Pirsiavash, 2024; Liu et al., 2024). Second,

*Corresponding Authors.

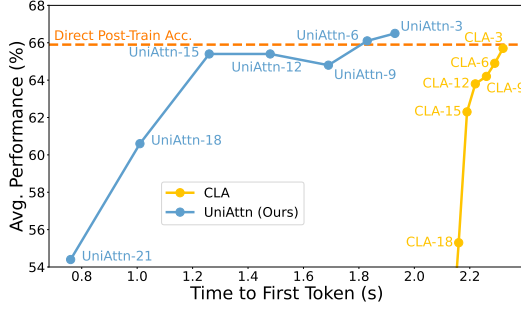


Figure 1: Comparisons of our UniAttn and directly applying cross-layer KV sharing (CLA) during post-training. “A-X” represents modifying total of X layers when applying A.

we also observe that *Softmax activations* exhibit high cross-layer similarity in the *top half layers* across various open-source pre-trained LLMs and post-training datasets. For instance, the average cross-layer similarity exceeds 0.9 on the top half layers in LLaMA-3.1 8B (Dubey et al., 2024) when evaluated on the Tulu3 (Lambert et al., 2024) training set (see Figure 2 for detailed results). This observation encourages us to reformulate efficient LLM design from the perspective of *Softmax redundancies*, and to exploit it during post-training.

In this paper, we propose to post-train LLMs with Softmax **Unification in Attention (UniAttn)**. Specifically, we firstly define “Superblock” as a group of consecutive LLM blocks in which each layer shares similar Softmax activations. We then propose to unify the Softmax activations of all blocks in each Superblock. Such unification can greatly reduce the K-cache size and softmax computations, thus significantly reducing memory and time costs. Additionally, to refine the approximation introduced by Softmax unification, we add a lightweight linear projection. With a simple and effective training scheme, this simple design effectively absorbs the approximation deviation induced by unification. To better interpret the performance gains of UniAttn, we present a theoretical analysis comparing it with existing cross-layer KV sharing methods. Our analysis reveals that UniAttn better preserves the benefits associated with model depth (Wang et al., 2024b) in deployed LLMs.

We conduct extensive experiments on four open-source pre-trained LLMs across two post-training scenarios: enhancing domain-specific capabilities and improving general capabilities. Our results show that UniAttn consistently achieves performance comparable to standard post-training while reducing inference costs. Compared to existing KV-

Model	FLOPs (%)			TTFT (%)	
	FFN	MHA (Softmax)		FFN	MHA (Softmax)
LLaMA-2 7B	66.8	33.2	(1.0)	0.2	99.8 (81.8)
LLaMA-3.1 8B	80.8	19.2	(0.9)	0.1	99.9 (83.9)
Mistral 7B	79.6	20.4	(0.9)	0.6	99.4 (80.1)
Gemma-2 9B	77.8	22.2	(1.0)	0.4	99.6 (79.1)

Table 1: Breakdown of FLOPs and TTFT time proportions (%) for major operations across different LLMs. Results are measured under a 8192 context length.

cache sharing baselines, UniAttn achieves better performance with $5\times$ more time reduction. Furthermore, our UniAttn can be combined with intra-layer KV sharing and KV-cache compression methods to further cut down the memory overhead, showing strong practicality for real-world applications.

Overall, we summarize our contribution as:

- We find that, although yielding high inference costs, Softmax activations in pre-trained LLMs exhibit significant redundancy, sharing high cross-layer similarities.
- We propose a *post-train* method, Softmax **Unification in Attention (UniAttn)**. Specifically, we define Superblocks to group LLM blocks with similar Softmax activations, and unify Softmax activations within each Superblock. Additionally, we leverage a lightweight linear projection to refine unification approximation. We also show theoretically that UniAttn better preserves model capabilities than cross-layer KV sharing.
- Extensive experiments show that UniAttn matches standard post-training performance while reducing inference costs, and is more robust than *retrofitting* cross-layer KV sharing during post-training.

2 Related Works

2.1 Post-Training LLMs

Building frontier LLMs for real-world applications involves two crucial stages: pre-training and post-training. Since pre-training data and methodologies are often proprietary, the research community has extensively explored post-training upon open-source pre-trained LLMs (Dubey et al., 2024; Rivière et al., 2024; Jiang et al., 2023) to enhance their *general* or *domain-specific* capabilities for deployment (Lian et al., 2024). Post-training is typically

performed on instruction-following or domain-specific datasets. Recently, various datasets have been curated to equip LLMs with specific abilities, including general instruction-following models (Teknum et al., 2024; Lambert et al., 2024; BAAI, 2024), medical QA models (Wu et al., 2023; Gururajan et al., 2024; Singhal et al., 2023), legal QA models (Huang et al., 2023; Zhou et al., 2024), and models with strong mathematical problem-solving capabilities (Liu and Low, 2023).

Existing research on post-training mainly focuses on creating specific datasets for equipping open-source LLMs with specific capabilities. Differently, we investigate the redundancies in pre-trained LLMs and leverage them for post-training inference-efficient LLMs.

2.2 KV-Cache Sharing Architectures

Unlike existing pruning and quantization approaches (Bolya et al., 2023; Xiong et al., 2024), efficient KV sharing architectures utilize structural redundancies to create inference-efficient model for deployment, and can be grouped into two categories: intra-layer KV sharing, including MQA (Shazeer, 2019) and GQA (Ainslie et al., 2023), and cross-layer KV sharing, including CLA (Brandon et al., 2024) and YOCO (Sun et al., 2024). Specifically, MQA (Shazeer, 2019) simplifies the attention mechanism by utilizing multiple query heads and a single KV head. GQA (Ainslie et al., 2023) takes a step further by organizing query heads as multiple groups and assigning different KV heads to each group. CLA (Brandon et al., 2024) proposes a cross-layer sharing mechanism to further reduce KV-cache memory overhead by sharing KV-cache across different layers. YOCO (Sun et al., 2024) transforms the original structure into self-decoders and cross-decoders, and adopts a global KV-cache across decoders.

Existing intra-layer KV sharing works have been adopted by various open-source LLMs (Dubey et al., 2024; Jiang et al., 2023; Rivière et al., 2024). Although cross-layer KV sharing introduces more aggressive memory reduction, according to our analysis in Section 1 and Section 3.3, it still suffers from efficiency and performance issues. To address this, we propose UniAttn, which achieves promising performance for post-training LLMs and outperforms competing methods.

3 Methodology

3.1 Preliminaries

In this paper, we focus on mainstream decoder-only LLM transformers. Each transformer block in an LLM features a multi-head self-attention (MHA) module and a feed-forward network (FFN). Given an input sequence $\mathbf{x} \in \mathbb{R}^{l \times d}$ where l denotes the sequence length and d denotes the hidden size, both MHA and FFN generate an output sequence with identical length and shape. We focus on the MHA module. Formally, we denote MHA output in layer i as $\mathbf{x}'_i$, where $\mathbf{x}'_i = \text{MHA}(\text{Norm}(\mathbf{x}_i)) + \mathbf{x}_i$. “Norm($\cdot$)” denotes the Pre-Norm, a component adopted by mainstream open-source LLMs (Dubey et al., 2024; Jiang et al., 2023; Rivière et al., 2024; Bai et al., 2023). In the MHA module in layer i , each token in the input sequence $\mathbf{x}_i$ is first projected by $W_{Q,i}$, $W_{K,i}$, and $W_{V,i}$, forming $Q_i, K_i \in d \times d_k$ and $V_i \in d \times d_v$. Then, the Softmax activation s_i is calculated by:

$$s_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right). \quad (1)$$

Subsequently, s_i is projected using V_i and the output weight matrix $W_{o,i}$ of the i -th layer, and the input to MHA is added through a residual connection to produce the MHA output $\mathbf{x}'_i$:

$$\mathbf{x}'_i = s_i V_i W_{o,i} + \mathbf{x}_i \quad (2)$$

3.2 UniAttn: Softmax Unification in Attention

Although existing KV-cache sharing methods, such as intra-layer and cross-layer KV sharing, can reduce the memory footprint, they offer limited inference time reduction (as shown in Figure 1). To achieve both memory-efficiency and inference acceleration, we investigate the Softmax operation because (1) calculating Softmax consumes a large portion of KV-cache with about 50% cache memory; and (2) Softmax operation leads to high inference latency, as validated by Table 1 and multiple studies (Geng et al., 2018; Koohpayegani and Pirsivash, 2024). To intuitively understand the redundancy of Softmax activations in LLMs, we conduct empirical analysis on the similarity of Softmax activations between layers. To ensure the generalizability of our study, we employ post-training datasets PMC (Wu et al., 2023), Infinity Instruct (Math) (BAAI, 2024), Tulu3 (Lambert et al., 2024), and long samples (F15K) provided by (Su, 2023), and utilize LLaMA-2 7B (Touvron et al., 2023), LLaMA-3.1

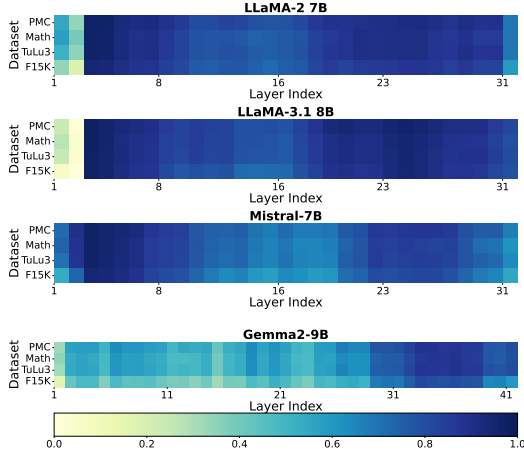


Figure 2: Cosine similarity results of average Softmax activations. Across all settings, the average Softmax activations of the top half of layers (i.e., the right half of each heatmap bar) share a high cosine similarity.

8B (Dubey et al., 2024), Mistral 7B (Jiang et al., 2023), and Gemma-2 9B (Rivière et al., 2024). We collect all Softmax activations, i.e., the s_i vectors in Equation (1), and compute the cosine similarity $\text{Sim}(i) = \cos(s_i, s_{i-1})$ for each sequence. We then average $\text{Sim}(i)$ over all sequences.

Observation: As shown in Figure 2, Softmax activations in *top half layers* share a high cross-layer similarity across different settings, showing that Softmax operations are highly redundant in LLMs. See Section C for quantitative analysis.

Motivated by such redundancy, we propose Softmax **Unification in Attention (UniAttn)** for post-training LLMs. Specifically, UniAttn groups consecutive decoder blocks in LLMs as “Superblocks”, and unifies the Softmax activations inside each Superblock to reduce the full Softmax operations and KV-cache size during inference. Additionally, to address the approximation induced by Softmax unification, we introduce a lightweight linear projection during post-training. UniAttn simultaneously reduces the GPU memory and inference latency while achieving competitive performance. Different from cross-layer KV sharing, UniAttn does not diminish the “depth factor”. We show this in Section 3.3.

Unify Softmax Activations in Superblocks. To apply UniAttn, we first define “Superblock” as a group of consecutive LLM blocks in which each layer shares Softmax activations. As shown in Figure 2, Superblocks can be created in the top half layers of many pre-trained LLMs. For simplicity, we create SuperBlocks of the same size. Inside

each SuperBlock, we only let the bottom block employ the Softmax operation. Other blocks simply re-use the Softmax activation matrix s calculated by the bottom block in the SuperBlock. Formally, for layer $i + b$ that re-use the Softmax activation s_i from layer i :

$$\mathbf{x}'_{i+b} = s_i V_{i+b} W_{o,i+b} + \mathbf{x}_{i+b} \quad (3)$$

Since only the bottom block contribute to calculating the Key projection, other blocks that reuse Softmax activations do not store their corresponding K-cache, leading to GPU memory-efficiency. Moreover, removing Softmax calculations in those layers also contribute to inference latency reduction. Similar to cross-layer KV sharing, UniAttn is also orthogonal to intra-layer KV sharing. In Section 4, we employ pre-trained LLMs with GQA to evaluate UniAttn.

Linear Compensation. Simply unifying Softmax activations unavoidably introduces approximation in the forward pass. Formally, the error ϵ in layer $i + b$ is:

$$\begin{aligned} \epsilon = \mathbf{x}_{i+b}^{\text{ori}} - \mathbf{x}_{i+b}^{\text{uni}} &= (\mathbf{x}_{i+b}^{\text{ori}} - \mathbf{x}_{i+b}^{\text{uni}}) \\ &+ (\text{softmax}(\frac{Q_{i+b} K_{i+b}^T}{\sqrt{d_k}}) - s_i) V_{i+b} W_{o,i+b} \end{aligned} \quad (4)$$

where $\mathbf{x}_{i+b}^{\text{ori}}$ and $\mathbf{x}_{i+b}^{\text{uni}}$ denote the input to layer $i + b$ in the original model and the model with UniAttn applied, respectively. According to (Koochpayegani and Pirsiavash, 2024), calculating linear projections yield *significantly lower* latency compared to the Softmax operation. And thus we study the effectiveness of compensating ϵ with a linear projection matrix W_c . To compensate for ϵ , we apply W_c to each layer with unified Softmax activations as:

$$\mathbf{x}'_{i+b} = s_i V_{i+b} W_{o,i+b} + \mathbf{x}_{i+b} + \mathbf{x}_{i+b} W_{c,i+b} \quad (5)$$

To find a proper initialization for W_c in a training-free manner, we propose to directly compensate ϵ :

Theorem 3.1. *The initialization for W_c that incurs minimal error when compensating $\mathbb{E}(\epsilon)$ satisfies:*

$$W_c = V \Sigma^+ U^T \mathbb{E}(\epsilon), \quad (6)$$

where $U \Sigma V^T$ denotes the SVD decomposition of $\mathbb{E}(\mathbf{x}_{i+b})$, Σ^+ denotes the pseudoinverse of Σ .

We leave the proof in the Appendix. In practice, we forward a small portion of training data

from the post-training datasets, and use the average values $\bar{\mathbf{x}}_{i+b}$ and $\bar{\epsilon}$ as estimations of $\mathbb{E}(\mathbf{x}_{i+b})$ and $\mathbb{E}(\epsilon)$. Considering that compensating for errors in previous layers would influence the calculation in subsequent layers, we calculate the initialization of W_c bottom-up. Experiments on the F15K dataset (Su, 2023) with sequence length 5,120 show that, after linear compensation, the approximation error ϵ in the top layer for LLaMA-2 7B and LLaMA-3.1 8B are reduced to 2.9% (382.87 $\rightarrow$ 11.09) and 6.4% (89.58 $\rightarrow$ 5.76) of the original errors without such compensation, respectively. This clearly demonstrates the effectiveness of the linear compensation. Please refer to Section I for more theoretical insights on the linear compensation.

To better train the linear transformations, we adopt a two-stage post-train pipeline. In the first stage, we only fine-tune the W_c weights *with early stop* and keep other weights frozen. We then conduct a full fine-tuning in the second stage. We formalize a detailed pipeline in Algorithm 1 in the Appendix.

3.3 Theoretical Analysis

Existing Methods Analysis. Existing KV-cache sharing methods can be mainly categorized as *intra-layer* and *cross-layer* KV sharing. Intra-layer KV sharing can be directly combined with UniAttn (see Section 4 for detailed results). While cross-layer KV sharing achieves a more aggressive memory reduction rate, through theoretical analysis, we show that it diminishes the impact of model depth on model activations through directly sharing the same KV-cache across layers. This negatively affects the capabilities of pre-trained LLMs, as model depth is a critical factor in determining their overall performance (Wang et al., 2024b).

Existing research (Razhigaev et al., 2024) has shown that LLM layers can be approximated as linear transformations. Together with the conclusion on training gradient dynamics (Xiong et al., 2020), we first give a reasonable assumption:

Assumption 3.2. In top layers of **pre-trained Pre-Norm LLMs**:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \delta_i, \quad \text{where} \quad \|\delta_i\| \ll \|\mathbf{x}_i\|. \quad (7)$$

We define δ as the “depth factor” that model applies on activations for further discussion. δ represents the extent to which model depth influences activations. Please refer to Section E for theoretical insights on proposing this assumption.

Analysis on Cross-layer KV Sharing: We adopt Assumption 3.2 to analyze the cross-layer KV sharing architecture, in which KV-cache is shared across multiple consecutive layers. Suppose layer $i + 1$ shares the KV-cache from layer i , the MHA operation in layer $i + 1$ can be written as:

$$\mathbf{x}'_{i+1} = \text{softmax}\left(\frac{QK_i^T}{\sqrt{d_k}}\right)V_iW_{o,i+1} + \mathbf{x}_{i+1}, \quad (8)$$

where K_i and V_i are the K and V matrices from layer i . We propose that:

Proposition 3.3. In **pre-trained Pre-Norm LLMs**, *cross-layer KV sharing diminishes the “depth factor” δ on activations.*

Please refer to Section G for detailed proof. Proposition 3.3 gives a rigorous proof showing that cross-layer KV sharing diminishes the impact of model depth on model activations. To offer an intuitive understanding of how KV sharing alters the structure of QKV computation within self-attention, we reformulate the operation as follows:

$$\begin{aligned} \text{QKV-Comp}_{i+1} &= \text{softmax}(A_i)V_i \\ &= \text{softmax}\left(\frac{\mathbf{x}_iW_{q,i+1}K_i^T}{\sqrt{d_k}}\right)V_i. \end{aligned} \quad (9)$$

Compared to QKV-comp_i in the previous layer, the only difference lies in the use of distinct W_q matrices, suggesting that the QKV computation in layer $i + 1$ can be viewed as introducing an additional query head applied to the same KV pairs as in layer i . In this sense, *cross-layer KV sharing implicitly transforms QKV computation in different layers into additional query heads.*

UniAttn Analysis. Our UniAttn reduces both memory cost and inference latency simultaneously. We underscore that unlike cross-layer KV sharing, our UniAttn does not diminish the “depth factor” δ in the forward pass. Please refer to Section H for detailed proof.

Proposition 3.4. In **pre-trained Pre-Norm LLMs**, *the “depth factor” in UniAttn is unignorable.*

4 Experiments

4.1 Experimental Settings

Post-Training Settings. We consider two post-training settings to validate our UniAttn: fine-tuning on domain-specific datasets and on general instruction following datasets. We choose PMC (Wu et al., 2023) as the domain-specific dataset

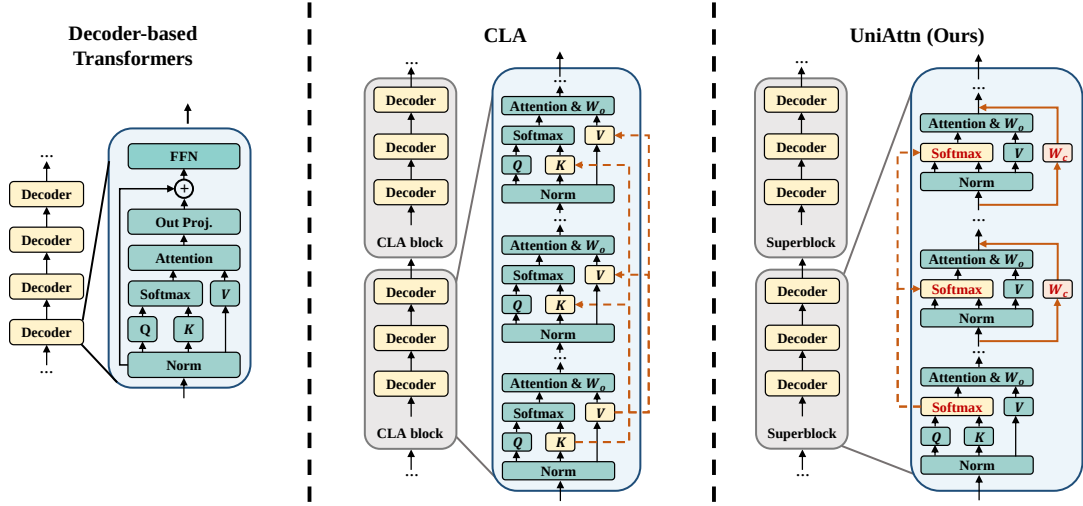


Figure 3: Pipeline comparison between standard decoder-based transformers, CLA (Brandon et al., 2024) (block size 3), and UniAttn (Superblock size 3). UniAttn shares the Softmax activations across layers in grouped Superblocks and adds linear transformation W_c to refine the unification approximation. For simplicity, only the self-attention calculation is illustrated for CLA and UniAttn.

and evaluate fine-tuned models on PubMedQA (Jin et al., 2019), MedMCQA (Pal et al., 2022), and MedQA (Jin et al., 2020). We choose Tulu3 (Lambert et al., 2024) as the general instruction following dataset and evaluate fine-tuned models on SIQA (Sap et al., 2019) and CommonsenseQA (Talmor et al., 2019). We employ (Gao et al., 2024) to perform those benchmark evaluations. Additionally, to further test whether UniAttn generalizes beyond standard short-context benchmarks, we conduct a long-context evaluation on the Niah Multikey task in RULER (Hsieh et al., 2024). Following the long-context training recipe of LongAlign (Bai et al., 2024), we post-train LLaMA-2 7B (as it does not possess long-context capabilities) and evaluate sequence lengths of 4,096, 8,192, 16,384, and 32,768. Those experiments are conducted with FlashAttention (Dao et al., 2022), which allows us to examine whether UniAttn is compatible with widely used acceleration kernels.

Model. We post-train 4 open-source pre-trained LLMs for evaluating our UniAttn: LLaMA-2 7B (Touvron et al., 2023), LLaMA-3.1 8B (Dubey et al., 2024), Mistral 7B (Jiang et al., 2023), and Gemma-2 9B (Rivière et al., 2024). *These models cover both non-GQA and GQA architectures, allowing us to evaluate whether UniAttn remains effective when intra-layer KV sharing has already been adopted.* We use base models that have undergone only pre-training.

Implementation details. We adopt the pattern

in Figure 2 and apply UniAttn in the *top half of layers*. Unless otherwise noted, we adopt Superblock size = 4, which yields a total of 4 Superblocks in LLaMA-2 7B, LLaMA-3.1 8B, Mistral 7B, and 5 Superblocks in Gemma-2 9B. For competing efficient architectures, we compare against LLMDrop (He et al., 2024) and CLA (Brandon et al., 2024). We evaluate two baseline configurations, denoted as HALF and FULL, corresponding to either matching the KV-cache retain rate of UniAttn or modifying the same set of layers as UniAttn, ensuring controlled comparisons in both memory reduction and operated-layer budget. For all experiments, we post-train LLMs for 1 epoch (Lambert et al., 2024). We adopt all training hyperparameter values based on existing research.

All training experiments are conducted on 8 H800 GPUs, covering four backbones, two post-training scenarios, multiple efficient-architecture baselines, TTFT profiling, end-to-end generation-time measurement, long-context evaluation with FlashAttention, KV-cache compression analysis, and Superblock ablations. Under this compute budget, we prioritize controlled comparisons across architectures and deployment settings.

Efficiency evaluation. We evaluate inference efficiency with multiple wall-clock metrics. First, we report time to first token (TTFT), which measures the latency of the prefill stage after receiving the input sequence. TTFT is measured on a single H800 GPU with context length 8,192. Second, we

Method	TTFT (s)	KV Cache	Medical				General		
			PubMedQa	MedMCQA	MedQA	AVG	SIQA	CommonsenseQA	AVG
LLama3.1-8B (w/ GQA)									
Pre-Trained	2.33	100%	75.0	56.8	59.1	63.6	46.8	71.7	59.3
Post-Train	2.33	100%	75.4	59.0	63.2	65.9	50.1	73.9	62.0
LLMDrop-Half	1.77	81.3%	75.6	<u>57.8</u>	<u>61.5</u>	<u>65.0</u>	51.4	<u>75.0</u>	<u>63.2</u>
LLMDrop-Full	1.47	62.5%	<u>78.2</u>	53.6	55.9	62.6	51.8	73.3	62.6
CLA-Half	2.29	81.3%	73.8	58.1	62.8	64.9	<u>51.7</u>	74.7	<u>63.2</u>
CLA-Full	2.22	62.5%	75.4	56.9	59.2	63.8	50.7	71.2	61.0
UniAttn (Ours)	<u>1.48</u>	81.3%	79.0	57.6	59.5	65.4	51.1	75.6	63.4
LLama2-7B (w/o GQA)									
Pre-Trained	2.20	100%	71.4	32.2	34.7	46.1	50.2	51.1	52.7
Post-Train	2.20	100%	75.4	48.8	49.2	57.8	51.6	66.3	59.0
LLMDrop-Half	1.74	81.3%	<u>75.2</u>	48.4	49.8	<u>57.8</u>	<u>51.7</u>	<u>66.7</u>	<u>59.2</u>
LLMDrop-Full	1.37	62.5%	75.0	<u>48.8</u>	48.8	57.5	51.2	65.2	58.2
CLA-Half	2.20	81.3%	74.4	48.4	50.6	<u>57.8</u>	51.3	66.1	58.7
CLA-Full	2.23	62.5%	<u>75.2</u>	45.7	47.5	56.1	50.2	65.6	57.9
UniAttn (Ours)	<u>1.44</u>	81.3%	75.6	49.4	<u>50.1</u>	58.4	51.9	67.7	59.8

Table 2: Post-training performance (%) comparison. **Bold** and underline denote the best and second-best performance of compressed models. We also report time to first token (TTFT) and KV-cache retain rate (KV Cache).

Method	Sequence Length				AVG
	4096	8192	16384	32768	
LLama2-7B (w/o GQA)					
Pre-Trained	82.67	0	0	0	20.67
Post-Train	<u>86.67</u>	87	80.33	72	81.5
LLMDrop-Half	87	83.67	77.67	68.67	79.25
CLA-Half	<u>86.67</u>	85.67	78.67	69.33	80.08
UniAttn (Ours)	87	<u>86.67</u>	<u>79.67</u>	<u>70.33</u>	<u>80.92</u>

Table 3: Post-training performance (%) comparison on Niah Multikey in RULER (Hsieh et al., 2024). **Bold** and underline denote the best and second-best performance.

evaluate end-to-end generation time by generating 1,024, 2,048, and 4,096 tokens from an 8,192-token context, showcasing the practical latency during autoregressive decoding. We additionally report the KV-cache retain rate to characterize memory usage.

4.2 Post-Training Performance Comparison

We verify the effectiveness of UniAttn on 2 open-source LLMs with different architectures, namely LLaMA-3.1 8B (with GQA) (Dubey et al., 2024) and LLaMA-2 7B (without GQA) (Touvron et al., 2023). We use LLMDrop (He et al., 2024) and CLA (Brandon et al., 2024) as competing efficient architectures, as both methods exploit the cross-layer redundancies in LLMs. For LLMDrop, we bypass the MHA modules with the highest input-output similarities in the top half of the layers. For CLA, we group the top half of the layers into CLA blocks, using the same configuration as in the

grouping of Superblocks. Since both competing methods drop the entire KV-cache in the operated layers, we compare two configurations: one with the same operating layers (X-Full) and another with the same KV-cache compression ratio (half operating layers, X-Half). We also conduct post-training after applying these methods.

We observe same conclusions on Mistral 7B (Jiang et al., 2023) and Gemma-2 9B (Rivière et al., 2024). See Section C for details.

Results. We compare different efficient architectures against directly post-training the original LLM (Post-Train) and using the pre-trained model (Pre-Trained) as baselines, as shown in Table 2. Among all competing architectures, our UniAttn achieves the best overall performance across model structures (with and without GQA) and post-training datasets. Besides significantly reducing inference costs in both time and memory, UniAttn maintains comparable performance to directly post-training the original model, well demonstrating its effectiveness. When comparing UniAttn with LLMDrop-Full, we observe that Softmax unification provides a similar acceleration rate to bypassing the entire MHA module, but it achieves significantly better post-training performance. This aligns with our finding that Softmax operations are both costly and redundant.

Analysis of CLA. Compared to CLA, UniAttn achieves better performance with significantly lower inference latency, demonstrating that it is

better suited for post-training than cross-layer KV sharing. Since CLA only reduces the latency associated with KV projections, its impact on TTFT is minimal. As a result, CLA-Half, CLA-Full, and the original model exhibit comparable TTFT times. Furthermore, CLA achieves comparable performance to LLMDrop when applied to the same number of layers. This suggests that the cross-layer attention mechanism employed by CLA is structurally equivalent to directly bypassing the MHA module. This observation supports our theoretical analysis that cross-layer KV sharing methods like CLA can diminish the depth factor in pre-trained LLMs, limiting their performance. Different from CLA, UniAttn avoids the issue of diminishing the depth factor and directly reduces time-costly Softmax computations, leading to better performance and lower inference latency.

Generalization to Long-Context Modeling. To further validate the effectiveness of UniAttn on long context modeling scenarios, we follow the training recipe of (Bai et al., 2024) and post-train LLaMA-2 7B, which lacks native support for extended contexts. We compare UniAttn with two strong baselines, CLA-Half and LLMDrop-Half, on the Niah Multikey task in the RULER benchmark (Hsieh et al., 2024). To successfully implement long-context modeling, here we conduct experiments with *FlashAttention* (Dao et al., 2022) applied. As shown in Table 3, UniAttn consistently ranks in the top two (second only to full post-training) and outperforms all competing methods, well demonstrating its strong generalization ability to complex long-context tasks. This also suggests that our UniAttn is compatible with common acceleration frameworks such as FlashAttention.

Generation Efficiency Analysis. To further analyze inference efficiency, we present the generation results of UniAttn against the comparing baselines, namely CLA-Half, CLA-Full, LLMDrop-Half, and LLMDrop-Full. We measure the end-to-end generation time with the LLaMA-3.1 8B backbone, and with generating 1024, 2048, and 4096 tokens using a context length of 8192. As shown in Figure 4, across all comparing methods, UniAttn consistently achieves Top-2 latency, only slightly behind LLMDrop-Full where all operated layers are simply skipped. These results demonstrate that unifying Softmax operations can substantially reduce the inference overhead during generation.

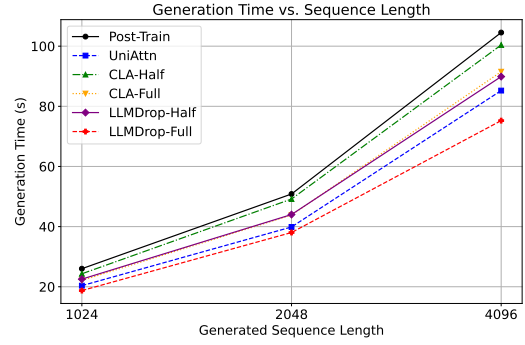


Figure 4: Generation efficiency comparison on LLaMA-3.1 8B with 8192 context length.

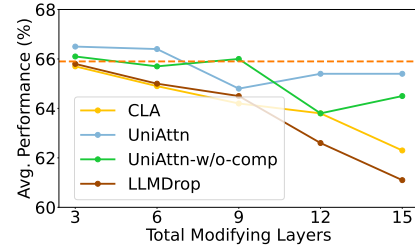


Figure 5: Comparison between baselines, UniAttn, and UniAttn without (“w/o”) linear compensation.

4.3 Experimental Analysis

We choose LLaMA-3.1 8B (Dubey et al., 2024) and the medical domain to conduct experimental analysis. See Section C for more analysis.

Ablation Studies. We do controlled experiments to evaluate the impact of each component in UniAttn, namely Softmax unification, linear compensation W_c , and the initialization for W_c . The results are shown in Table 4. We observe that after unifying the Softmax activations, both adding linear compensation and using proper initialization contribute positively to the final performance. While linear compensation consistently improves performance across different initialization methods, appropriate initialization proves critical for effectively fine-tuning W_c .

Plain Softmax Unification v.s. Baselines. To further substantiate the theoretical analysis and demonstrate the effectiveness of Softmax unification, we compare UniAttn (without linear compensation, UniAttn-w/o-comp) against CLA and LLMDrop. As shown in Figure 5, simply unifying Softmax activations yields notable improvements over baselines, highlighting its core contribution. The linear projection serves as an enhancement, bringing UniAttn closer to post-train-level results.

Further Inference Cost Reduction. To further maximize the potential of UniAttn, we integrate it with the KV-cache compression method

Method	Unify Softmax	Compensation	Init W_c	PubMed	MedMCQA	MedQA	AVG
Post-Train	-	-	-	75.4	59.0	63.2	65.9
UniAttn	✓	×	×	77.0	55.7	<u>58.8</u>	63.8
	✓	✓	zero-init	78.0	55.8	58.1	63.9
	✓	✓	Theorem 3.1	<u>78.2</u>	<u>57.1</u>	58.4	64.6
	✓	✓	Theorem 3.1 + fine-tune	79.0	57.6	59.5	65.4

Table 4: Ablation study results. **Bold** and underline denote the best and second-best UniAttn performance.

Method	TTFT (s)	H ₂ O Comp.	KV-Cache	AVG
Post-Train	2.33	-	100%	65.9
CLA-Half	2.29	-	81.3%	64.9
CLA-Full	2.22	-	62.5%	63.8
UniAttn	1.48	-	81.3%	65.4
	1.48	20%	65.0%	65.2
UniAttn+H ₂ O	1.46	40%	48.8%	65.0
	1.46	60%	32.5%	63.3

Table 5: Results of inference costs analysis. “H₂O Comp.” denotes the compression rate introduced by H₂O and “KV-Cache” denotes the KV-Cache retain rate.

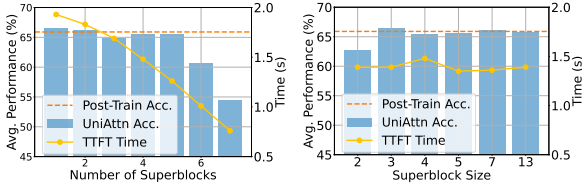


Figure 6: Hyperparameter analysis results on average accuracy (%) and TTFT latency (s). Left: Different total numbers of grouped Superblocks (each with size 4). Right: Different Superblock sizes (total of 12 layers that utilize unified Softmax activation).

H₂O (Zhang et al., 2023) to further reduce memory overhead by keeping only the essential and the recent tokens in the KV-cache. We report the TTFT latency and the total KV-cache retain rate. As shown in Table 5, UniAttn achieves a significant reduction in TTFT latency compared to CLA under the same KV-cache retain rate. When combined with H₂O, UniAttn+H₂O reduces the KV-cache retain rate to below 50% while maintaining an average performance of 65.0%, achieving both better performance, lower inference latency, and higher KV-cache compression rate than CLA. These results clearly demonstrate the inference efficiency of UniAttn, highlighting its substantial benefits for real-world deployment scenarios.

Hyperparameter Analysis. We perform a hyperparameter analysis on the number and the size of Superblocks to evaluate their impact on latency and post-training performance. First, we examine the effect of varying the number of Superblocks. Using a consistent Superblock size of 4, we group

different numbers of Superblocks from the top layers to the bottom layers. For instance, grouping 2 Superblocks indicates that layer 25-28 and 29-32 form the two Superblocks. The results are shown in Figure 6 (left). From the perspective of TTFT latency, increasing the number of Superblocks significantly reduces latency. From the perspective of performance, while grouping certain top layers as Superblocks does not substantially affect model performance, continuing to increase the number of Superblocks leads to sharp performance drops. Second, we analyze the impact of Superblock size. For fair comparison, we maintain *a consistent total number of layers that utilize unified Softmax activations from preceding layers* as 12. Since a Superblock with size b unifies Softmax in $b - 1$ layers, we set b such that $b - 1$ is a factor of 12, resulting in Superblock sizes of 2, 3, 4, 5, 7, and 13, and number of Superblocks as 12, 6, 4, 3, 2, and 1, respectively. As shown in Figure 6 (right), the size of the Superblocks does not significantly affect post-training performance. *For simplicity, we use a Superblock size of 4*, although fine-grained tuning of Superblock size could further improve post-training performance.

5 Conclusion

In this work, we explored LLM redundancies and efficient architectures in the context of post-training. While KV-cache sharing methods successfully reduce the memory overhead, they yield limited inference time reduction and degrade post-training performance. To address this, we investigated the primary bottleneck in LLMs, the Softmax operation, and discovered significant redundancies across various pre-trained LLMs. Based on these findings, we proposed UniAttn, an efficient architecture that unifies Softmax activations across layers in post-training. Experiments on a series of settings validate the effectiveness and efficiency of UniAttn. Our UniAttn is particularly well-suited for real-world deployments, enjoying faster inference, less memory cost, and better performance.

Limitations

Currently, UniAttn is designed for post-training existing pre-trained LLMs. We leave its extensions to other training stages for future works.

References

- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. [GQA: training generalized multi-query transformer models from multi-head checkpoints](#). 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.
- BAAI. 2024. [Infinity instruct](#).
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. [LongAlign: A recipe for long context alignment of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1376–1395, Miami, Florida, USA. Association for Computational Linguistics.
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. [Token merging: Your vit but faster](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. [Reducing transformer key-value cache size with cross-layer attention](#). *CoRR*, abs/2405.12981.
- Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. [An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models](#). In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXI*, volume 15139 of *Lecture Notes in Computer Science*, pages 19–35. Springer.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- John Fearnley and Rahul Savani. 2015. The complexity of the simplex method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 201–208.
- 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, and 5 others. 2024. [A framework for few-shot language model evaluation](#).
- Xue Geng, Jie Lin, Bin Zhao, Anmin Kong, Mohamed M. Sabry Aly, and Vijay Chandrasekhar. 2018. [Hardware-aware softmax approximation for deep neural networks](#). In *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part IV*, volume 11364 of *Lecture Notes in Computer Science*, pages 107–122. Springer.
- Ashwin Kumar Gururajan, Enrique Lopez-Cuena, Jordi Bayarri-Planas, Adrian Tormos, Daniel Hinjos, Pablo Bernabeu-Perez, Anna Arias-Duart, Pablo Agustin Martin-Torres, Lucia Urcelay-Ganzabal, Marta Gonzalez-Mallo, Sergio Alvarez-Napagao, Eduard Ayguadé-Parra, and Ulises Cortés Dario Garcia-Gasulla. 2024. [Aloe: A family of fine-tuned open healthcare llms](#). *Preprint*, arXiv:2405.01886.
- Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024. [What matters in transformers? not all attention is needed](#). *CoRR*, abs/2406.15786.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. [Ruler: What’s the real context size of your long-context language models?](#) *arXiv preprint arXiv:2404.06654*.
- Quzhe Huang, Mingxu Tao, Chen Zhang, Zhenwei An, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023. [Lawyer llama technical report](#). *Preprint*, arXiv:2305.15062.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *CoRR*, abs/2310.06825.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. [What disease does this patient have? A large-scale open domain question answering dataset from medical exams](#). *CoRR*, abs/2009.13081.

- Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. [Pubmedqa: A dataset for biomedical research question answering](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2567–2577. Association for Computational Linguistics.
- Soroush Abbasi Koohpayegani and Hamed Pirsiavash. 2024. [Sima: Simple softmax-free attention for vision transformers](#). In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2024, Waikoloa, HI, USA, January 3-8, 2024*, pages 2595–2605. IEEE.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. [Tulu 3: Pushing frontiers in open language model post-training](#). *CoRR*, abs/2411.15124.
- Haoran Lian, Junmin Chen, Wei Huang, Yizhe Xiong, Wenping Hu, Guiguang Ding, Hui Chen, Jianwei Niu, Zijia Lin, Fuzheng Zhang, and Di Zhang. 2024. [Breaking the stage barrier: A novel single-stage approach to long context extension for large language models](#). *Preprint*, arXiv:2412.07171.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. [Visual instruction tuning](#). 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*.
- Shiwei Liu, Guanchen Tao, Yifei Zou, Derek Chow, Zichen Fan, Kauna Lei, Bangfei Pan, Dennis Sylvester, Gregory Kielian, and Mehdi Saligane. 2024. [Consmax: Hardware-friendly alternative softmax with learnable parameters](#). In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9.
- Tiedong Liu and Bryan Kian Hsiang Low. 2023. [Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks](#). *Preprint*, arXiv:2305.14201.
- Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. [Shortgpt: Layers in large language models are more redundant than you expect](#). *CoRR*, abs/2403.03853.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- Ankit Pal, Logesh Kumar Umapathi, and Malaikanan Sankarasubbu. 2022. [Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering](#). In *Conference on Health, Inference, and Learning, CHIL 2022, 7-8 April 2022, Virtual Event*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.
- Anton Razzhigaev, Matvey Mikhalechuk, Elizaveta Goncharova, Nikolai Gerasimenko, Ivan Oseledets, Denis Dimitrov, and Andrey Kuznetsov. 2024. [Your transformer is secretly linear](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5376–5384, Bangkok, Thailand. Association for Computational Linguistics.
- Morgane Rivi re, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L onard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram , Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, and 80 others. 2024. [Gemma 2: Improving open language models at a practical size](#). *CoRR*, abs/2408.00118.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. [Socialliqa: Commonsense reasoning about social interactions](#). *CoRR*, abs/1904.09728.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Noam Shazeer. 2019. [Fast transformer decoding: One write-head is all you need](#). *Preprint*, arXiv:1911.02150.
- Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaeckermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, and 12 others. 2023. [Towards expert-level medical question answering with large language models](#). *Preprint*, arXiv:2305.09617.
- Jianlin Su. 2023. Rectified rotary position embeddings. <https://github.com/bojone/rerope>.
- 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](#). *CoRR*, abs/2405.05254.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. [Commonsenseqa: A question answering challenge targeting commonsense knowledge](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.

Ryan Teknium, Jeffrey Quesnelle, and Chen Guang. 2024. [Hermes 3 technical report](#). *Preprint*, arXiv:2408.11857.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ao Wang, Hui Chen, Jianchao Tan, Kefeng Zhang, Xunliang Cai, Zijia Lin, Jungong Han, and Guiguang Ding. 2024a. [Prefixkv: Adaptive prefix kv cache is what vision instruction-following models need for efficient generation](#). *Preprint*, arXiv:2412.03409.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2024b. [Deepnet: Scaling transformers to 1,000 layers](#). *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(10):6761–6774.

Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. [Pmc-llama: Further finetuning llama on medical papers](#). *CoRR*, abs/2304.14454.

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.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. [On layer normalization in the transformer architecture](#). 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 10524–10533. PMLR.

Yizhe Xiong, Hui Chen, Tianxiang Hao, Zijia Lin, Jungong Han, Yuesong Zhang, Guoxin Wang, Yongjun Bao, and Guiguang Ding. 2024. [PYRA: parallel yielding re-activation for training-inference efficient task adaptation](#). In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part IX*, volume 15067 of *Lecture Notes in Computer Science*, pages 455–473. Springer.

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.

Zhi Zhou, Jiang-Xin Shi, Peng-Xiao Song, Xiao-Wen Yang, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. 2024. [Lawgpt: A chinese legal knowledge-enhanced large language model](#). *Preprint*, arXiv:2406.04614.

A License for Scientific Artifacts

The Training dataset PMC (Wu et al., 2023) is licensed under OpenRAIL license¹. The Training dataset TuLu3 (Lambert et al., 2024) is licensed under Open Data Commons License². The Mistral model (Jiang et al., 2023), Gemma-2 model (Rivière et al., 2024), LLaMA-2 (Huang et al., 2023) model, and LLaMA-3.1 model (Dubey et al., 2024) are licensed under Apache License 2.0³. The LongAlign evaluation dataset is also licensed under Apache License 2.0⁴. The evaluation datasets (Hsieh et al., 2024; Talmor et al., 2019; Sap et al., 2019; Jin et al., 2020; Pal et al., 2022; Jin et al., 2019) are subject to the MIT license⁵. All usages of scientific artifacts in this paper obey the corresponding licenses.

B More Implementation Details for Reproducibility

Post-Training Datasets. Regarding the medical domain, we directly employ the instruction tuning dataset for PMC⁶ (Wu et al., 2023) (i.e., training stage-2 for PMC-LLaMA) for post-training. The dataset consists of 513,999 instruction-input-output pairs. Regarding the general instruction tuning scenario, we employ the Tulu 3 SFT Mixture⁷ and filter for data with only 1 round of conversation, resulting in 896,070 input-output pairs.

Training Hyperparameters. The training hyperparameters for all experiments are reported in table Table 6, Table 7 and Table 8. Note that we adopt the learning rate and weight decay values according to existing research, in which those pre-trained models are post-trained under reported schedules.

Evaluation Details. For all experiments, we evaluate the final checkpoint after post-training for

¹<https://www.licenses.ai/blog/2022/8/26/bigscience-open-rail-m-license>

²<https://opendatacommons.org/licenses/>

³<https://choosealicense.com/licenses/apache-2.0/>

⁴<https://choosealicense.com/licenses/apache-2.0/>

⁵<https://choosealicense.com/licenses/mit/>

⁶https://huggingface.co/datasets/axiong/pmc_llama_instructions

⁷<https://huggingface.co/datasets/allenai/tulu-3-sft-mixture>

Method	Model	Learning Rate	Weight Decay	Batch Size	Epochs	Superblock Size	Superblock Groups
UniAttn	LLaMA-2 7B	2e-5	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	LLaMA-3.1 8B	7e-6	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	Mistral 7B	1e-6	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	Gemma-2 9B	1e-6	0	48	1	4	[22-25], [26-29], [30-33], [34-37], [38-41]

Table 6: Training hyperparameter details of UniAttn.

Method	Model	Learning Rate	Weight Decay	Batch Size	Epochs	CLA Block Size	CLA Block Groups
CLA-Half	LLaMA-2 7B	2e-5	0	48	1	4	[25-28], [29-32]
	LLaMA-3.1 8B	7e-6	0	48	1	4	[25-28], [29-32]
	Mistral 7B	1e-6	0	48	1	4	[25-28], [29-32]
	Gemma-2 9B	1e-6	0	48	1	4	[30-33], [34-37], [38-41]
CLA-Full	LLaMA-2 7B	2e-5	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	LLaMA-3.1 8B	7e-6	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	Mistral 7B	1e-6	0	48	1	4	[17-20], [21-24], [25-28], [29-32]
	Gemma-2 9B	1e-6	0	48	1	4	[22-25], [26-29], [30-33], [34-37], [38-41]

Table 7: Training hyperparameter details of CLA.

benchmark evaluation. Regarding the medical domain, we report the 0-shot result on both PubMed, MegMCQA, and MedQA datasets. Regarding the general instruction tuning scenario, we report 0-shot result on both SIQA and CommonsenseQA for LLaMA-3.1 8B, Mistral 7B, and Gemma-2 9B models. For the earlier-released LLaMA-2 7B model, since it has a shorter pre-training context length, we report its 5-shot result as Tulu 3 mainly enhances model performance with longer context.

Prompt for Post-Training. We employ the same prompt for post-training in both the medical domain and general instruction tuning scenario:

```
"Below is an instruction that
describes a task, paired with an input
that provides further context. Write a
response that appropriately completes
the request.\n\n###
Instruction:\n{instruction}\n\n###
Input:\n{input}\n\n###
Response:{output}<EOS_TOKEN>"
```

We simply keep the instruction field empty if no instruction is provided.

Experiment Packages. We report the version numbers of used packages in Table 9.

C More Experiment Results

Main Results on Mistral 7B and Gemma-2 9B.

We adopt the settings in Table 2 and conduct the same experiments on Mistral 7B (Jiang et al., 2023) and Gemma-2 9B (Rivière et al., 2024). The results are shown in Table 10. Consistent with the results on the LLaMA models, our UniAttn achieves the best overall performance on both LLMs and

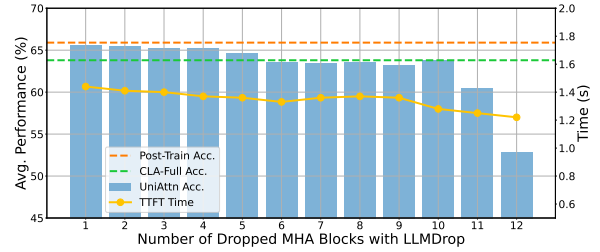


Figure 7: Results on further dropping UniAttn layers (%). We present the average accuracy and time to first token (TTFT) latency under different number of dropped MHA blocks via LLMDrop. Note that we only drop MHA blocks with unified Softmax activations.

post-training datasets while significantly reducing both time and memory costs. When operating the same number of layers, UniAttn achieves similar TTFT time to LLMDrop, well demonstrating the effectiveness of Softmax activation unification. Although CLA and LLMDrop achieve competitive results in some settings (e.g., CLA-Half in post-training Gemma on general instruction datasets), *they cannot provide consistent performance across different settings*. Additionally, even on the Mistral and Gemma models, CLA achieves similar performance to its corresponding LLMDrop setting, again indicating that CLA diminishes the depth factor in pre-trained LLMs.

Quantitative Analysis of Softmax Redundancies. In Figure 2 of the main article, we observed that Softmax activations in *top half layers* share a high cross-layer similarity across different models and datasets. To provide a more comprehensive analysis, here we take LLaMA-3.1 8B as an ex-

Method	Model	Learning Rate	Weight Decay	Batch Size	Epochs	# of Dropped Layers	Index of Dropped Layers
LLMDrop-Half	LLaMA-2 7B	2e-5	0	48	1	6	23,20,32,27,22,24
	LLaMA-3.1 8B	7e-6	0	48	1	6	19,22,20,21,27,23
	Mistral 7B	1e-6	0	48	1	6	17,32,31,21,22,24
	Gemma-2 9B	1e-6	0	48	1	8	29,26,21,30,27,40,38,31
LLMDrop-Full	LLaMA-2 7B	2e-5	0	48	1	12	23,20,32,27,22,24,31,29,25,30,28,26
	LLaMA-3.1 8B	7e-6	0	48	1	12	19,22,20,21,27,23,30,29,28,26,24,25
	Mistral 7B	1e-6	0	48	1	12	17,32,31,21,22,24,29,23,28,25,27,26
	Gemma-2 9B	1e-6	0	48	1	15	29,26,21,30,27,40,38,31,32,33,35,39,36,37,34

Table 8: Training hyperparameter details of LLMDrop. The index of dropped layers are ordered by input-output similarity.

Package	Version	Package	Version
PyTorch	2.0.0	transformers	4.46.0
deepspeed	0.10.0	tokenizers	0.20.1
datasets	2.14.3		

Table 9: Versions of used packages.

ample and present some statistics of the cosine similarity of Softmax activations. As shown in Table 11, cosine similarities appear to be consistent across different datasets, with average values ranging from 0.79 to 0.83. The maximum cosine similarity remains consistently high at 0.97, indicating that the overall range of cosine similarity remains stable across layers and datasets. The differences for cosine similarity between the top and bottom halves of layers indicate that Softmax operations are more redundant in deeper layers, as noted in the main article. The standard deviation values suggest that redundancy is fairly consistent across datasets. Overall, these results indicate that Softmax activations for pre-trained LLMs exhibit high redundancies.

Empirical Statistics for the “Depth Factor”. To further support Assumption 3.2 and the claim made in Proposition 3.3, we applied cross-layer sharing to the top layers of LLaMA-3.1 8B and evaluated 100 randomly selected samples from the PMC post-training corpus. The average results of the following terms are: $||\delta_i||/||\mathbf{x}_i|| = 0.070$, $||J(A_i)|| = 0.041$, and $||J(A_i)\hat{\delta}||/||\text{Softmax}(A_i)|| = 0.038$. These results support that (1) δ_i is small comparing to $||\mathbf{x}_i||$, validating Assumption 3.5; (2) the depth factor is indeed diminished under cross-layer KV-cache sharing, consistent with Proposition 3.6.

Superblock Grouping Strategy. To further investigate other grouping strategies, we consider a redundancy-aware approach and use a similarity-based adaptive scheme. Specifically, we assigned Superblocks by iteratively grouping consecutive

layers with the highest average cosine similarity of Softmax activations. As shown in Table 13, under the same total number of “grouped layers”, this similarity-based grouping variant achieved comparable performance to the default fixed-size Superblock setting, indicating that the performance of UniAttn is robust to different grouping strategies, and that adaptive grouping does not yield meaningful gains over the simpler fixed-size design.

Different Compensation Designs. To compare different compensation designs, we attach the input and output of the W_c transformation to different activations. As shown in Table 12, directly compensating the MHA output with its input yields the best results. When adopting “finer compensation granularities”, i.e., positioning the input and output closer together (Rows 3 and 4), post-training performance decreases. Similarly, adopting “coarser compensation granularities”, i.e., moving the output compensation to after the FFN (Row 2), also results in lower performance. These findings suggest that using a linear transformation to compensate for errors specifically within the MHA module is the best-performing approach.

Experimental Analysis on the Impact of Model Depth. In Section 3.3, we demonstrated that, unlike cross-layer KV sharing methods, UniAttn does not diminish the depth factor in pre-trained LLMs. To provide experimental evidence for this, we further apply LLMDrop to the MHA modules that utilize unified Softmax activations from preceding layers (a total of 12 layers in UniAttn for the LLaMA-3.1 8B model) in post-trained UniAttn models. Specifically, we compute the input-output similarities of these MHA modules and simply prune the top k MHA modules with the highest similarities. The pruned model is then directly evaluated without additional fine-tuning. The average performance w.r.t. k is shown in Figure 7. As shown in the figure, pruning a few MHA modules from our UniAttn model still results in higher

Method	TTFT (s)	KV Cache	Medical				General		
			PubMedQa	MedMCQA	MedQA	AVG	SIQA	CommonsenseQA	AVG
Mistral-7B (w/ GQA)									
Pre-Trained	1.94	100%	75.8	48.3	50.9	58.3	46.7	56.5	51.6
Post-Train	1.94	100%	78.8	49.7	60.3	62.9	53.7	76.6	65.2
LLMDrop-Half	1.54	81.3%	78.0	50.6	<u>59.0</u>	62.5	<u>52.5</u>	74.5	63.5
LLMDrop-Full	1.21	62.5%	79.0	50.2	<u>56.7</u>	62.0	51.8	73.6	62.7
CLA-Half	1.94	81.3%	78.6	<u>56.0</u>	58.5	<u>64.4</u>	52.0	77.9	<u>65.0</u>
CLA-Full	1.91	62.5%	<u>78.8</u>	55.0	55.0	62.9	50.5	75.1	62.8
UniAttn	1.23	81.3%	78.2	57.3	60.5	65.3	53.4	<u>77.5</u>	65.5
Gemma2-9B (w/ GQA)									
Pre-Trained	2.23	100%	78.6	57.6	60.6	65.6	51.5	68.4	60.0
Post-Train	2.23	100%	78.6	58.6	61.9	66.4	54.4	75.6	65.0
LLMDrop-Half	1.94	81.0%	78.6	56.4	<u>60.5</u>	<u>65.2</u>	55.6	68.4	62.0
LLMDrop-Full	1.66	64.3%	78.2	54.8	58.7	63.9	<u>54.8</u>	62.9	58.9
CLA-Half	2.19	78.6%	79.2	<u>56.6</u>	56.3	64.0	53.5	76.0	64.8
CLA-Full	2.24	64.3%	74.0	39.3	38.5	50.6	48.3	51.8	43.4
UniAttn	1.69	82.1%	<u>79.0</u>	60.7	64.3	68.0	53.7	<u>74.4</u>	<u>64.1</u>

Table 10: Post-training performance comparison. **Bold** and underline denote the best and second-best performance of compressed models. For each method, we report their time to first token (TTFT, in seconds) and KV-cache retain rate (KV Cache).

Dataset	Average	Difference Between Top Half and Bottom Half Layers	Max	Min	Standard Deviation
PMC	0.8347	0.1314	0.9729	0.0159	0.1955
Math	0.8243	0.1149	0.9734	0.0384	0.1860
Tulu3	0.8214	0.1212	0.9719	0.0307	0.1896
F15K	0.7927	0.1475	0.9710	0.0033	0.2109

Table 11: Statistics of the cosine similarity of Softmax activations for LLaMA-3.1 8B across different datasets.

Input to W_c	Output adds to	AVG
MHA input	MHA output	65.4
MHA input	FFN output	64.1
MHA input	Activation before W_o	<u>65.1</u>
Activation projected by W_v	Activation before W_o	64.3

Table 12: Performance comparison (%) of different compensation designs.

Method	Grouping	PubMed	MedMCQA	MedQA	AVG
Post-Train	-	75.4	59.0	63.2	65.9
UniAttn	Fixed size (4)	79.0	57.6	59.5	65.4
	Adaptive	78.8	57.3	59.5	65.2

Table 13: Performance (%) comparison of different Superblock grouping methods in UniAttn for LLaMA-3.1 8B. **Bold** denotes the best performance of compressed models.

performance than the CLA-Full model variant. After pruning 10 MHA modules, UniAttn+LLMDrop achieves an average performance of 63.8, matching the performance of CLA-Full reported in Table 2. This demonstrates that UniAttn better preserves the impact of model depth compared to CLA. Furthermore, we can also conclude that the depth of the model significantly contributes to the post-train performance, thereby validating our theoretical analysis.

D Pipeline for Applying UniAttn during Post-Training

We provide a detailed pipeline for applying UniAttn as Algorithm 1.

E Theoretical Insights for Proposing Assumption 3.2

We first give a series of conclusions (namely Lemma E.1, Proposition E.2, Proposition E.3, and Proposition E.4) as preliminaries.

Lemma E.1. (A linear Pre-Norm system has bounded growth) $f_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $i \in \mathbb{N}$ are LINEAR transformations and $\text{Norm}(\cdot)$ yields a vector

Algorithm 1 UniAttn Pipeline

Input: Pre-trained Model M , Post-train dataset D , SuperBlock size b , Apply SuperBlock merging from layer i_s to i_e
 Create UniAttn model M_{uni} by creating $\frac{i_e - i_s + 1}{b}$ SuperBlocks in M
 Sample a subset D_{init} with 1000 samples from D
 For model M , forward all samples from D_{init} , calculate the average output activations $\mathbf{x}'_i$ for each MHA block.
for $i = 1$ **to** $\frac{i_e - i_s + 1}{b}$ **do**
 for $j = 1$ **to** $b - 1$ **do**
 For model M_{uni} , forward all samples from D_{init} , calculate the average output activations $\mathbf{x}'_{\text{uni}, i_s + (i-1)b + j}$ and the average input activations $\mathbf{x}_{\text{uni}, i_s + (i-1)b + j}$ for MHA block in layer $i_s + (i-1)b + j$.
 Calculate the initialization for the $W_{c, i_s + (i-1)b + j}$ matrix in layer $i_s + (i-1)b + j$ according to Theorem 3.1.
 Insert the initialized $W_{c, i_s + (i-1)b + j}$ matrix to layer $i_s + (i-1)b + j$ in model M_{uni} .
end for
end for
 Freeze all training parameters other than the W_c matrices in M_{uni} . Train W_c with dataset D and apply early stop when loss stops decreasing (we consider the exponential moving average of training loss).
 Conduct full fine-tuning on M_{uni} with D .
Output: Trained UniAttn model M_{uni} .

with unit Frobenius-norm (denoted as $\|\cdot\|$). Let $\mathbf{x}_0 \in \mathbb{R}^n$ be the input to the system. Consider an L -layer Pre-Norm architecture defined by:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + f_k(\text{Norm}(\mathbf{x}_{k-1})), \quad (10)$$

for $k = 1, 2, \dots, L$.

If the largest singular value in all transformation matrices is bounded by λ , then:

$$\|\mathbf{x}_L\| \leq \|\mathbf{x}_0\| + \lambda L \quad (11)$$

Proof. Unroll the expression of $\mathbf{x}_L$ and inductively:

$$\mathbf{x}_L = \mathbf{x}_0 + \sum_{k=1}^L F_k(\text{Norm}(\mathbf{x}_{k-1})) \quad (12)$$

As $\|\text{Norm}(\mathbf{x}_k)\| = 1$ for all k , then:

$$\begin{aligned} \|\mathbf{x}_L\| &= \|\mathbf{x}_0 + \sum_{k=1}^L F_k(\text{Norm}(\mathbf{x}_{k-1}))\| \\ &\leq \|\mathbf{x}_0\| + \sum_{k=1}^L \|F_k(\text{Norm}(\mathbf{x}_{k-1}))\| \\ &\leq \|\mathbf{x}_0\| + \sum_{k=1}^L \|\lambda \cdot \text{Norm}(\mathbf{x}_{k-1})\| \\ &= \|\mathbf{x}_0\| + \lambda L \end{aligned} \quad (13)$$

□

Lemma E.1 shows that a Pre-Norm linear system has bounded growth. The norm of output by each layer grows at most linearly with depth L .

Proposition E.2. (Pre-trained LLMs are linear Pre-Norm systems) Pre-trained decoder-based LLMs exhibit a high linearity score. Formally, let $A, B \in \mathbb{R}^{n \times d}$ denote the normalized input and output of a decoder block in LLM, respectively,

$$\min_X \|AX - B\| \approx 1 \quad (14)$$

Proposition E.2 has been validated by (Razhigaev et al., 2024) on pre-trained LLMs, suggesting that we can approximate each decoder block by a linear transformation with bounded singular values, thus placing pre-trained LLMs structurally “close” to a linear Pre-Norm system as in Lemma E.1. In the following discussion, we approximate pre-trained LLMs with an equivalent linear Pre-Norm system and investigate its input and output in each layer.

Proposition E.3. (Pre-trained LLM layers primarily change the magnitude of activations) The input and output of a layer in pre-trained decoder-based LLMs share a high cosine similarity.

Proposition E.3 has been validated by (Men et al., 2024) on pre-trained LLMs, indicating that each layer in LLMs predominantly change the magnitude of the activation. We leverage the following conclusion for analyzing the change of activation magnitude.

Proposition E.4. (Top layers of Pre-Norm transformers have smaller singular values) Training Pre-Norm transformers leads to top layers receiving smaller updates (gradient norms decrease from bottom to top).

(Xiong et al., 2020) has given both a theoretical proof sketch and experimental evidence of Proposition E.4. Based on Proposition E.2, we approximate each layer in LLMs as a linear transformation. To further show that the approximated linear transformations of the top layers in Pre-Norm LLMs tend to have smaller singular values, we give the following proof:

Proof. With Lemma E.1 and Proposition E.2, we have shown that pre-trained Pre-Norm LLMs can be approximated as linear Pre-Norm systems. Suppose a pre-trained LLM operates on activations $\mathbf{x} \in \mathbb{R}^d$, each layer i in the pre-trained LLM can be treated as a linear transformation matrix $W_i \in \mathbb{R}^{d \times d}$. A generic gradient-based update can be expressed as:

$$W_i^{t+1} = W_i^0 - \sum_{t=0}^t \eta \nabla_{W_i} \mathcal{L}^t \quad (15)$$

Hence, the norm between initial and final weights satisfies:

$$\|W_i^{t+1} - W_i^0\| \leq \sum_{t=0}^t \|\eta \nabla_{W_i} \mathcal{L}^t\| \quad (16)$$

We denote the singular value decomposition of $W_i^{t+1} - W_i^0$ as $W_i^{t+1} - W_i^0 = U \Sigma V^T$, it is easy to derive that:

$$\begin{aligned} \|W_i^{t+1} - W_i^0\| &= \sqrt{\text{tr}[(\|W_i^{t+1} - W_i^0\|)(\|W_i^{t+1} - W_i^0\|)^T]} \\ &= \sqrt{\text{tr}(U \Sigma V^T V \Sigma^T U^T)} \\ &= \sqrt{\text{tr}(\Sigma \Sigma^T)} = \sqrt{\sum_{i=1}^d \sigma_i^2} \end{aligned} \quad (17)$$

where σ_i denote the i -th singular value in Σ . Normally, the largest singular value $\sigma_{\max}$ dominates in the quadratic term, thus we can further write:

$$\|W_i^{t+1} - W_i^0\| = \sqrt{\sum_{i=1}^d \sigma_i^2} \approx \sigma_{\max}(W_i^{t+1} - W_i^0) \quad (18)$$

Based on the basic features of singular values, we can show that:

$$\begin{aligned} \sigma_{\max}(W_i^{t+1}) &\leq \sigma_{\max}(W_i^0) + \sigma_{\max}(W_i^{t+1} - W_i^0) \\ &\leq \sum_{t=0}^t \|\eta \nabla_{W_i} \mathcal{L}^t\| + \sigma_{\max}(W_i^0) \end{aligned} \quad (19)$$

v	N/A	1	2	3	4	8	16
PPL	5.914	2.890	2.871	2.835	2.850	2.822	2.837

Table 14: Perplexity results for initializations with different v values on LLaMA-3.1 8B in the medical domain. ‘‘N/A’’ represents applying Softmax Unification without linear projections.

Due to having smaller values of $\|\eta \nabla_{W_i} \mathcal{L}^t\|$ (i.e., smaller gradient norm), compared to earlier layers, the top layers typically have smaller updates on the largest singular value, thus tend to have smaller $\sigma_{\max}$ values. $\square$

Since pre-trained LLMs mainly change the magnitude of the activation (as shown in Proposition E.3), we can use the size of the largest singular values in each layer as an indicator of the corresponding activation magnitude change. In Pre-Norm architectures, each layer operates on an input with a constant norm. This leads to the assumption that the top-layer transformations generally generate outputs with smaller norms, which is Assumption 3.2 we propose in the article.

F Proof for Theorem 3.1

Proof. Solving for the initialization of W_c that incurs minimal compensation error is equivalent to solving the following optimization problem:

$$\min_{W_c} \|\mathbb{E}(\mathbf{x}_{i+b} W_c - \epsilon)\|. \quad (20)$$

Since we use a small subset of the training dataset (with S samples, we denote this as the *initialization set*) to calculate the initialization, the objective above can be rewritten as:

$$\min_{W_c} \frac{1}{S} \sum_{j=1}^S \|\mathbf{x}_{i+b,j} W_c - \epsilon_j\|. \quad (21)$$

This is a Least Absolute Deviations problem, and can be solved with the simplex algorithm, which is NP as shown in (Fearnley and Savani, 2015). To guarantee that the initialization method is efficient even in the worst case, we in turn consider the Least Squares variant:

$$\min_{W_c} \frac{1}{S} \sum_{j=1}^S \|\mathbf{x}_{i+b,j} W_c - \epsilon_j\|^2. \quad (22)$$

The Least Squares variant is a convex optimization problem, and has a closed-form solution as follows:

$$W_c = X^+ E, \quad (23)$$

where X is a matrix created by concatenating all $\mathbf{x}_{i+b,j}$ instances, E is created by concatenating all ϵ_j instances, and X^+ represents the Moore-Penrose inverse of X . When actually calculating for W_c , it is time-consuming to calculate X^+ as the size of X is big. Therefore, we set the *maximum instance size* of X as v . Specifically, we average $\mathbf{x}$'s and ϵ 's for every $\frac{S}{v}$ instances in the initialization set, and calculate W_c on the averaged $\mathbf{x}$'s and ϵ 's. For simplicity, we denote the averaged $\mathbf{x}$'s and ϵ 's as $\mathbb{E}(\mathbf{x}_{i+b})$ and $\mathbb{E}(\epsilon)$. Consequently, the calculation for W_c is formulated as:

$$W_c = V\Sigma^+U^T\mathbb{E}(\epsilon), \quad (24)$$

where $V\Sigma^+U^T$ is the Moore-Penrose inverse of $\mathbb{E}(\mathbf{x}_{i+b})$, calculated via SVD. $\square$

To find an appropriate v value for initialization, we have conducted preliminary experiments on LLaMA-3.1 8B in the medical domain. Specifically, we have benchmarked the perplexity on the *initialization set* after inserting all linear projections with initialized weights calculated. As shown in Table 14, most performance contribution comes from the act of adding initialization, and increasing the number of v affects the PPL values slightly. These experiments guided us to adopt $v = 1$ for our experiments due to its simplicity.

G Proof for Proposition 3.3

Proof. For simplicity, we suppose $\mathbf{x}_{i+1}$ is already normalized by the Pre-Norm. We can re-write Equation (8) as:

$$\begin{aligned} \mathbf{x}'_{i+1} &= \text{softmax}\left(\frac{\mathbf{x}_{i+1}W_{q,i+1}K_i^T}{\sqrt{d_k}}\right)V_iW_{o,i+1} \\ &\quad + \mathbf{x}_{i+1} \\ &= \text{softmax}\left(\frac{(\mathbf{x}_i + \delta_i)W_{q,i+1}K_i^T}{\sqrt{d_k}}\right)V_iW_{o,i+1} \\ &\quad + \mathbf{x}_{i+1} \\ &= \text{softmax}\left(\frac{\mathbf{x}_iW_{q,i+1}K_i^T}{\sqrt{d_k}}\right. \\ &\quad \left.+ \frac{\delta_iW_{q,i+1}K_i^T}{\sqrt{d_k}}\right)V_iW_{o,i+1} + \mathbf{x}_{i+1} \end{aligned} \quad (25)$$

Let $A_i = \frac{\mathbf{x}_iW_{q,i+1}K_i^T}{\sqrt{d_k}}$, $\hat{\delta}_i = \frac{\delta_iW_{q,i+1}K_i^T}{\sqrt{d_k}}$. By expanding Softmax to the first order:

$$\begin{aligned} \mathbf{x}'_{i+1} &= [\text{softmax}(A_i) + J(A_i)\hat{\delta}_i]V_iW_{o,i+1} \\ &\quad + \mathbf{x}_{i+1} + o(\hat{\delta}_i^2), \end{aligned} \quad (26)$$

where $J(A_i)$ is the Jacobian of Softmax at A_i . We ignore the $o(\hat{\delta}_i^2)$ remainder. (Xiao et al., 2024) has discovered that the pre-trained LLMs generally exhibit an ‘‘attention sink’’ feature, in which self-attentions heavily attend to the ‘‘sink token’’ at the beginning of a sequence. We adopt such feature to estimate $\|J(A_i)\|$. According to the statistical pattern concluded by (Xiao et al., 2024), attention logits $\mathbf{a} \in \mathbb{R}^l$ of a sequence with length l is approximately:

$$a_i = 1 \ (0 < i < d), \ a_j = -1 \ (i \geq d), \ d \ll l \quad (27)$$

(Xiao et al., 2024) empirically adopts $d = 4$. For a sequence of $l = 1024$, $\|J(\text{softmax}(\mathbf{a}))\| \approx 0.03 \ll 1$, and decreases as l increases. Therefore, we can conclude that the coefficient of the $\hat{\delta}$ term is significantly smaller than that of the Softmax term, making the contribution of the ‘‘depth factor’’ negligible. $\square$

H Proof for Proposition 3.4

Proof. In the forward pass of UniAttn:

$$\begin{aligned} \mathbf{x}'_{i+1} &= s_i(\mathbf{x}_{i+1}W_{v,i+1})W_{o,i+1} + \mathbf{x}_{i+1} \\ &= s_i(\mathbf{x}_i + \delta_i)W_{v,i+1}W_{o,i+1} + \mathbf{x}_{i+1} \end{aligned} \quad (28)$$

Observing that LLM layers primarily change activation magnitudes (see Proposition E.3), we assume $\mathbf{x}_i$ and δ sharing similar directions, further leading to $\frac{\|s_i\mathbf{x}_iW_{v,i+1}W_{o,i+1}\|}{\|s_i\delta_iW_{v,i+1}W_{o,i+1}\|} \approx \frac{\|\mathbf{x}_i\|}{\|\delta_i\|}$. Self-attention in UniAttn maintains the relative magnitude of the depth factor, ensuring its influence on the output. $\square$

I More Insights on Linear Compensation

To demonstrate the effectiveness of our linear compensation strategy, we propose the following theorem:

Theorem I.1. $A, B \in \mathbb{R}^{m \times n}$, $X \in \mathbb{R}^{n \times n}$. Suppose that each element from A and B are drawn from a Gaussian distribution such that $a_{ij}, b_{ij} \sim N(0, 1)$. It satisfies that:

$$\mathbb{E}\left[\min_X \|AX - B\|\right] = \begin{cases} \sqrt{n(m-n)}, & \text{if } m \geq n, \\ 0, & \text{if } m < n. \end{cases} \quad (29)$$

See the following subsections for the proof. In the article, we apply the F15K dataset (Su, 2023) to calculate the error ϵ on LLaMA-2 7B (Touvron et al., 2023) and LLaMA-3.1 8B (Dubey et al.,

2024) models after inserting the initialized linear transformation W_c according to Theorem 3.1. While the sequence length being $m = 5120$ and the hidden size of applied models being $n = 4096$, the error is 11.09 and 5.76 after linear compensation, correspondingly, which are both *magnitudes lower* than their expectations on random data. Those results demonstrate the effectiveness of our linear compensation strategy.

We give the proof for Theorem I.1 in the following subsections.

I.1 Preliminaries

To prove Theorem I.1, we adopt the notations in Theorem I.1 and prove a series of lemmas first.

Lemma I.2. *Let $P = AA^+$, P is an orthogonal projection and $\text{Im}(P) = \text{Col}(A)$.*

Proof. The Moore-Penrose inverse matrix exhibits some basic features, namely $AA^+A = A$ and $(AA^+)^T = AA^+$. Using that feature we can easily verify that:

$$P^2 = AA^+AA^+ = (AA^+A)A^+ = AA^+ = P \quad (30)$$

$$P^T = (AA^+)^T = AA^+ = P \quad (31)$$

So P is an orthogonal projection.

Next, we prove $\text{Im}(P) = \text{Col}(A)$. For any vector $\mathbf{v} \in \text{Col}(A)$, there exists a $\mathbf{c} \in \mathbb{R}^n$ that $\mathbf{v} = A\mathbf{c}$. We have:

$$P\mathbf{v} = AA^+A\mathbf{c} = (AA^+A)\mathbf{c} = A\mathbf{c} = \mathbf{v} \quad (32)$$

So $\mathbf{v} \in \text{Im}(P)$. Therefore, $\text{Col}(A) \subseteq \text{Im}(P)$. Since $P = AA^+$, it is obvious that $\text{Im}(P) \subseteq \text{Col}(A)$. Hence, we can conclude that $\text{Im}(P) = \text{Col}(A)$. $\square$

Lemma I.3. *$I - P$ is an orthogonal projection and $\text{Im}(I - P) = \text{Col}(A)^\perp$.*

Proof. First, we prove that $I - P$ is an orthogonal projection:

$$(I - P)^2 = I - 2P + P^2 = I - 2P + P = I - P \quad (33)$$

$$(I - P)^T = I - P^T = I - P \quad (34)$$

Then, we prove that $\text{Im}(I - P) = \text{Col}(A)^\perp$. For any $\mathbf{v} \in \text{Im}(I - P)$, there exists a $\mathbf{c} \in \mathbb{R}^m$ that $\mathbf{v} = (I - P)\mathbf{c}$. Let $\mathbf{d} \in \text{Col}(A)$, then there exists

a $\mathbf{w} \in \mathbb{R}^n$ that $\mathbf{d} = A\mathbf{w}$. Computing the inner product of $\mathbf{v}$ and $\mathbf{d}$ yields:

$$\begin{aligned} \langle \mathbf{v}, \mathbf{d} \rangle &= \langle (I - P)\mathbf{c}, A\mathbf{w} \rangle = \langle \mathbf{c}, (I - P)A\mathbf{w} \rangle \\ &= \langle \mathbf{c}, (A - AA^+A)\mathbf{w} \rangle = \langle \mathbf{c}, 0 \rangle = 0 \end{aligned} \quad (35)$$

This shows that $\mathbf{v} \in \text{Col}(A)^\perp$, and in turn $\text{Im}(I - P) \subseteq \text{Col}(A)^\perp$.

Conversely, suppose $\mathbf{u} \in \text{Col}(A)^\perp$. Recall Lemma I.2 that P is an orthogonal projection and $\text{Im}(P) = \text{Col}(A)$, we have $P\mathbf{u} = 0$. Therefore,

$$(I - P)\mathbf{u} = \mathbf{u} - P\mathbf{u} = \mathbf{u} \quad (36)$$

This shows that $\mathbf{u} \in \text{Im}(I - P)$, and in turn $\text{Col}(A)^\perp \subseteq \text{Im}(I - P)$. Therefore, $\text{Im}(I - P) = \text{Col}(A)^\perp$. $\square$

From Lemma I.3, we can immediately conclude that:

Corollary I.4. *$\text{rank}(I - P) = m - \min(m, n)$*

Proof. $\text{rank}(I - P) = \dim \text{Im}(I - P) = \dim \text{Col}(A)^\perp = m - \text{rank}(A)$. Recall Theorem I.1 that every element from A is sampled from a Gaussian distribution, so that by probability of 1 we have $\text{rank}(A) = \min(m, n)$, which leads to the conclusion of this corollary. $\square$

Lastly, we give a lemma on orthogonal projections to a Gaussian-sampled vector:

Lemma I.5. *If $\mathbf{b} \sim N(\mathbf{0}, I_{m \times m})$ is a standard Gaussian in $\mathbb{R}^m$ and Q is an orthogonal projector of rank k , then*

$$\mathbb{E}[\|Q\mathbf{b}\|] = \sqrt{k} \quad (37)$$

Proof. $\mathbb{E}[\|Q\mathbf{b}\|^2] = \mathbb{E}[\mathbf{b}^T Q^T Q \mathbf{b}] = \mathbb{E}[\mathbf{b}^T Q \mathbf{b}] = \mathbb{E}[\text{tr}(\mathbf{b}^T Q \mathbf{b})] = \mathbb{E}[\text{tr}(Q \mathbf{b} \mathbf{b}^T)] = \text{tr}(Q \mathbb{E}[\mathbf{b} \mathbf{b}^T])$. As $\mathbf{b} \sim N(\mathbf{0}, I_{m \times m})$, $\mathbb{E}[\mathbf{b} \mathbf{b}^T] = I$. Therefore, using the trace property of projectors:

$$\text{tr}(Q \mathbb{E}[\mathbf{b} \mathbf{b}^T]) = \text{tr}(Q) = k \quad (38)$$

In turn, we can conclude that $\mathbb{E}[\|Q\mathbf{b}\|] = \sqrt{k}$. $\square$

I.2 Proof

Finally, we give the proof for Theorem I.1.

Proof. According to Theorem 3.1, for given A and B , the optimal solution X^* that satisfies $\|AX^* - B\| = \min_X \|AX - B\|$ is given by:

$$X^* = A^+B, \quad (39)$$

where A^+ is the Moore-Penrose inverse of A . We can re-write the objective as:

$$\begin{aligned}
\mathbb{E}(\|AA^+B - B\|^2) &= \mathbb{E}(\|(I - AA^+)(-B)\|^2) \\
&= \mathbb{E}(\|(I - AA^+)B\|^2) \\
&= \sum_{i=1}^n \mathbb{E}(\|(I - AA^+)\mathbf{b}_i\|^2)
\end{aligned} \tag{40}$$

According to Lemma I.3 and Corollary I.4, $I - AA^+$ is an orthogonal projection and $\text{rank}(I - AA^+) = m - \min(m, n)$. With Lemma I.5, we know that for any i , $\mathbb{E}(\|(I - AA^+)\mathbf{b}_i\|^2) = m - \min(m, n)$. Therefore, we can conclude that:

$$\mathbb{E}(\|AA^+B - B\|^2) = \begin{cases} n(m - n), & \text{if } m \geq n, \\ 0, & \text{if } m < n. \end{cases} \tag{41}$$

which leads to the conclusion in Theorem I.1. $\square$