PRUNE&COMP: Free Lunch for Layer-Pruned LLMs via Iterative Pruning with Magnitude Compensation

Xinrui Chen[1], Hongxing Zhang[2], Fanyi Zeng[1], Yongxian Wei[1], Yizhi Wang[1], Xitong Ling[1], Guanghao Li[1], Chun Yuan[1]*

[1] Shenzhen International Graduate School, Tsinghua University
[2] School of Information Science and Technology, Guangdong University of Foreign Studies
cxr22@tsinghua.org.cn, yuanc@sz.tsinghua.edu.cn

Abstract

Layer pruning is a viable technique for compressing large language models while achieving acceleration proportional to the pruning ratio. In this work, we identify that removing any layer induces a magnitude gap in hidden states, and demonstrate that a simple compensation operation leads to superior performance in iterative layer pruning. This key observation motivates us to propose PRUNE&COMP, a novel, plug-and-play iterative layer pruning scheme that leverages magnitude compensation to mitigate such gaps in a training-free manner. Specifically, we first estimate the magnitude gap of layer removal and then eliminate it by rescaling the remaining weights offline. We further demonstrate the advantages of PRUNE&COMP in improving the stability of iterative pruning. When integrated with an iterative prune-and-compensate loop, PRUNE&COMP consistently enhances existing layer pruning metrics. For instance, when 5 layers of LLaMA-3-8B are pruned with the prevalent Taylor+ metric, PRUNE&COMP reduces PPL from 512.78 to 16.34 and retains 90.57% of the original performance across 9 question-answering tasks, outperforming the baseline by 24.72%.

Code —
https://github.com/chenxinrui-tsinghua/PruneAndComp

Introduction

In recent years, large language models (LLMs) have achieved remarkable success across a wide range of natural-language-processing tasks (Achiam et al. 2023; Jiang et al. 2023; Team 2025; Dubey et al. 2024; Team et al. 2025; Liu et al. 2024; Guo et al. 2025). As model size increases, LLMs exhibit substantial performance gains; however, the vast parameter count incurs prohibitive computational costs and long inference latency. The model compression community has thus introduced effective model compression schemes, primarily quantization, knowledge distillation, and pruning (Sreenivas et al. 2024; Muralidharan et al. 2024; Sun et al. 2024; Ashkboos et al. 2024; van der Ouderaa et al. 2023; Xia et al. 2023; Sarah et al. 2024; Hu et al. 2024; Chen et al. 2024b; Li et al. 2025; Chen et al. 2023). Among these, pruning is an auspicious approach that reduces model size by removing unimportant parameters or components.

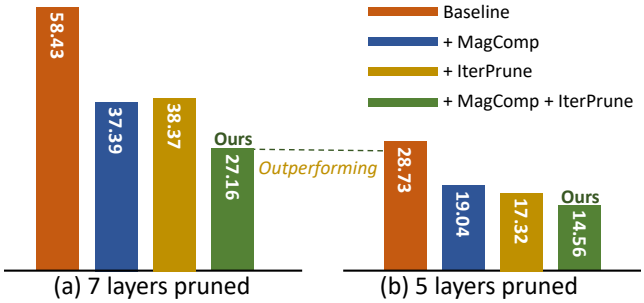

Figure 1: Average perplexity (↓) of pruned LLaMA-3-8B on WikiText-2, C4, and PTB. Our 7-layer-pruned model outperforms the 5-layer-pruned baseline. Baseline: naive pruning using CosSim(BI) metric (Men et al. 2024); + MagComp: magnitude compensation; + IterPrune: iterative pruning.

Structured pruning is the mainstream pruning paradigm. Unlike semi-structured sparsity or unstructured pruning, which irregularly eliminate weights or modules and therefore introduce irregular memory access, structured pruning avoids reliance on specialized hardware or software optimizations and delivers genuine acceleration. Within the structured pruning literature, the dominant techniques fall into depth pruning and width pruning. Width pruning removes unimportant weight channels and attention heads, thereby shrinking the width of LLMs. Depth pruning, also known as layer pruning, discards entire Transformer layers to reduce model depth. Existing studies (Gromov et al. 2024; Kim et al. 2024) show that, at moderate pruning ratios, layer pruning can retain most of the original performance without retraining. Moreover, by reducing model depth, layer pruning shortens LLM inference latency and achieves higher speed-ups at the same pruning ratio, without requiring hardware- or software-specific support.

The majority of layer pruning methods (Gromov et al. 2024; Kim et al. 2024; Song et al. 2024; Chen et al. 2024a; Men et al. 2024) have designed sophisticated layer importance metrics to locate redundant layers. Representative works based on layer-output similarity (Men et al. 2024; Chen et al. 2024a; Gromov et al. 2024), performance-impact metrics (Kim et al. 2024; Song et al. 2024), and gradient-based metrics (Kim et al. 2024; Ma, Fang, and Wang 2023)

*Corresponding author.

have attracted considerable attention due to their strong empirical results. These methods typically identify and remove redundant layers in either a one-shot or iterative strategy. Iterative methods account for inter-layer dependencies and generally yield more competitive performance (Kim et al. 2024; Song et al. 2024).

Nevertheless, even though existing layer pruning strategies preserve most performance without retraining, they still suffer from noticeable performance degradation. We investigate this phenomenon and offer two key observations:
(1) We demonstrate that removing any layer from an LLM introduces a significant gap in the magnitudes of hidden states, as shown in Figure 2. This is an intrinsic property of LLMs and is independent of the pruning metric. Our intuition is to compensate for this magnitude gap, showing that a simple compensation strategy can easily restore model performance, as presented in Figure 1.
(2) Although iterative layer pruning methods account for inter-layer dependencies, we hypothesize that the already-damaged pruned model negatively affects subsequent layer importance assessments, thus degrading performance. To verify this hypothesis, we propose maintaining the activation magnitude of the model during iterative pruning, expecting that this allows the pruned model to better identify truly redundant layers. As presented in Figure 1, we show that a simple compensation operation during the iterative process significantly mitigates the performance degradation and thus achieves superior results.

Building on these observations, we introduce a simple yet effective method, PRUNE&COMP, which compensates for the magnitude gap caused by layer pruning via weight modifications that incur no online inference overhead. When combined with iterative layer pruning, PRUNE&COMP improves the robustness of the pruned model and thus enhances the effectiveness of the iterative layer importance estimation. Our major contributions are as follows:

- We reveal that layer removal introduces magnitude gap, causing instability and performance drop in iterative layer pruning. This degradation is rooted in the intrinsic properties of the model, regardless of the pruning metric.
- We introduce PRUNE&COMP, a training-free recipe that couples iterative layer pruning with magnitude compensation to close the gap in a training-free manner.
- Extensive experiments show that PRUNE&COMP consistently boosts prevalent pruning metrics by a large margin, while adding no online inference overhead.

Related Works

Width Pruning

Width pruning for LLMs aims to reduce the computational footprint by selectively removing redundant or less informative channels, attention heads, and their coupled structures. LLM-Pruner (Ma, Fang, and Wang 2023), the first structured pruning framework tailored for LLMs, leverages gradient-based importance estimation to identify and remove non-essential coupled structures, achieving substantial compression while preserving core model capabilities. Sheared LLaMA (Xia et al. 2023) accelerates LLM pre-training by incorporating targeted structured pruning and dynamic batch loading, demonstrating improved training efficiency and outperforming baseline models of similar size. Wanda (Sun et al. 2023) introduces a retraining-free technique that induces sparsity by eliminating weights with the smallest product of magnitude and input hidden states, offering a simple yet effective approach to parameter reduction. Similarly, FLAP (An et al. 2024) exploits activation fluctuations to a novel retraining-free structured pruning framework that enhances storage efficiency and inference speed.

Depth Pruning

Depth pruning, commonly known as layer pruning, is an alternative to width pruning that compresses LLMs by removing complete Transformer blocks. The key advantage of this approach is that it maintains the parameter dimensions of the remaining layers. At the same pruning rate, layer pruning achieves higher speed-ups without additional architectural dependencies. ShortGPT (Men et al. 2024) proposes Block Influence (BI), a layer-level importance metric that captures the semantic shift between a layer's input and output representations. LLM-Streamline (Chen et al. 2024a) leverages cosine similarity to assess layer importance and introduces a novel compression metric, stability, to quantify the structural sensitivity of models. SLEB (Song et al. 2024) exploits the redundancy across adjacent transformer blocks based on block-level Perplexity (PPL) evaluations. Shortened LLaMA (Kim et al. 2024) adopts a straightforward depth pruning approach, guided by gradient-based and magnitude-based metrics to determine removable layers.

Motivation

Preliminaries on LLM Layer Pruning

LLMs are predominantly built upon the Transformer architecture, which comprises a stack of Transformer decoder layers with residual connections. We denote the ℓ-th Transformer layer as $f(X^{(\ell)}, \theta^{(\ell)})$, where $X^{(\ell)}$ represents its input hidden states and $\theta^{(\ell)}$ signifies its corresponding parameters. Given the prevalent use of the pre-norm architecture in LLMs, the output of the ℓ-th layer can be expressed as:

$$X^{(\ell+1)} = X^{(\ell)} + f(X^{(\ell)}, \theta^{(\ell)}). \tag{1}$$

To remove LLM layers with index from ℓ^*-th layer to $(\ell^* + n)$-th layer, we skip these layers and pass the input of the ℓ^*-th layer to $(\ell^* + n)$-th layer, i.e.,

$$X^{(\ell^*+n)} = X^{(\ell^*)} + f(X^{(\ell^*)}, \theta^{(\ell^*+n)}). \tag{2}$$

Especially, $n = 1$ indicates the removal of the ℓ^*-th layer.

Layer Pruning Produces Magnitude Gap

We quantitatively analyze the magnitude gain introduced by every individual LLM layer. For each layer ℓ, we calculate the channel-wise averaged magnitude gain ratio between its input and output hidden states over a calibration set. The magnitude gain ratio of layer ℓ is defined as:

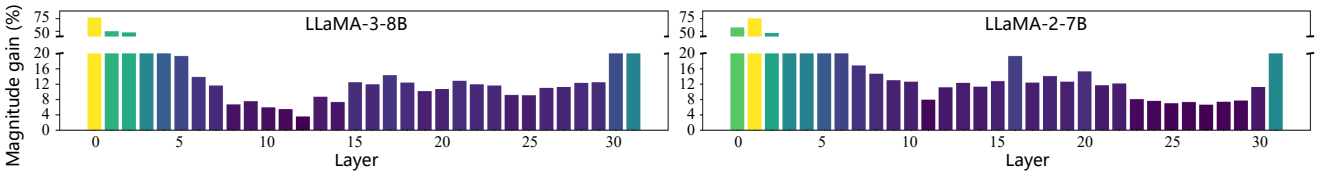

Figure 2: Visualization on the channel-wise averaged magnitude gain ratio of each layer. All layers produce magnitude gain.

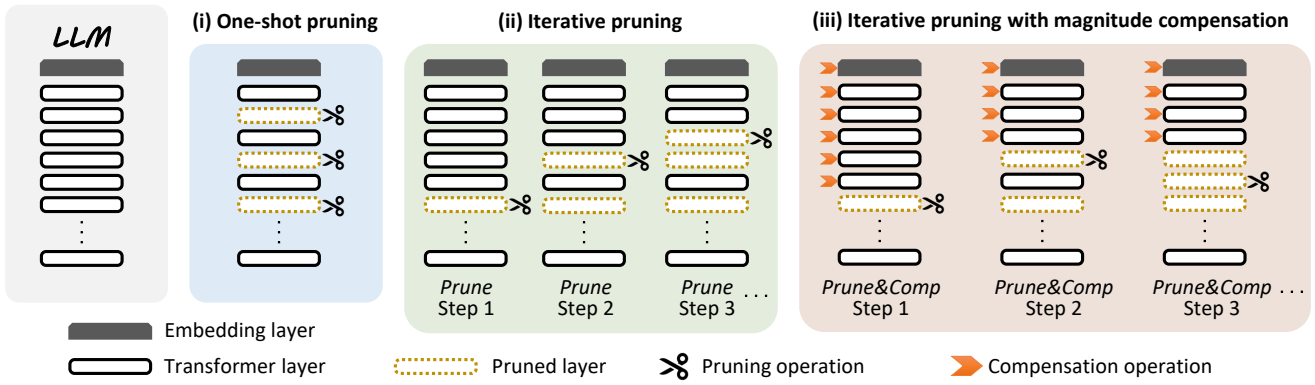

Figure 3: Comparison on conventional pruning strategy and the proposed PRUNE&COMP.

$$\delta^{(\ell)} = \left(\mathbb{E}_{(X^{(\ell)}, X^{(\ell+1)}) \in \mathcal{D}} \frac{1}{C} \sum_{k}^{C} \frac{\|X_{:,k}^{(\ell+1)}\|_1}{\|X_{:,k}^{(\ell)}\|_1} - 1 \right) \times 100\%, \tag{3}$$

where $\mathcal{D}$ denotes the calibration set, and $X^{(\ell)} \in \mathbb{R}^{B \times T \times C}$ represents the input hidden state of the ℓ-th layer with batch size B, length of tokens T and hidden dimension C, obtained through calibration samples. Figure 2 visualizes the individual layer magnitude gain ratio of LLaMA-3-8B and LLaMA-2-7B. It is observed that each layer produces a pronounced increase in magnitude, and the largest surge exceeds 70% and occurs in the early layers. Additional visualizations across more models refer to the Appendix. Consequently, removing any layer inevitably creates a corresponding magnitude gap, regardless of the pruning metrics. Our intuition is to compensate for this gap and preserve the scale of hidden states. As shown in Figure 1, simply compensating this magnitude gap (+MagComp) boosts the performance.

Rethinking Iterative Layer Pruning

Figure 3 provides a detailed comparison diagram of one-shot pruning, iterative pruning, and the proposed PRUNE&COMP scheme. One-shot pruning makes its entire pruning decision in a single pass, offering high search efficiency. However, removing multiple layers simultaneously ignores inter-layer dependencies, and the cumulative impact can be far more destructive than the sum of the individual effects. Iterative pruning, by contrast, re-evaluates the layer importance after every removal. This procedure captures second-order interactions and generally yields a superior accuracy-compression trade-off (Kim et al. 2024; Song et al. 2024). Nevertheless, each iterative pruning step damages the model, which in turn hampers robust assessment of the remaining layers. Motivated by this, we propose to compensate the pruned model during the iterative process, enabling it to locate redundant layers more precisely and thus producing the best performance as shown in Figure 1.

Methods

Layer Pruning Metrics

In this work, the following five prevalent layer pruning metrics are employed to identify redundant layers in LLMs.

Cosine Similarity-based Metrics.

- **CosSim(BI)** (Men et al. 2024): Block Influence (BI) measures the importance of each layer by evaluating the similarity between a layer's input and its output. The BI score for the ℓ-th layer can be calculated as follows:

$$BI_\ell = \mathbb{E}_{X,t} \frac{(X_t^{(\ell)})^T X_t^{(\ell+1)}}{||X_t^{(\ell)}||_2 ||X_t^{(\ell+1)}||_2}, \tag{4}$$

  where $X_t^{(\ell)}$ denotes the hidden states from the t-th token position of the ℓ-th layer's input hidden states. A higher BI score indicates a high cosine similarity between $X^{(\ell)}$ and $X^{(\ell+1)}$, suggesting redundancy of layer ℓ.

- **CosSim(CL)** (Chen et al. 2024a; Gromov et al. 2024): Similarly, a series of contiguous layers (CL) with high cosine similarity between their input and output indicates redundancy. When pruning n layers, the importance of n consecutive layers from layer ℓ to layer $\ell + n$ can be calculated as:

$$CL_{\ell,n} = \mathbb{E}_{X,t} \frac{(X_t^{(\ell)})^T X_t^{(\ell+n)}}{||X_t^{(\ell)}||_2 ||X_t^{(\ell+n)}||_2}, \tag{5}$$

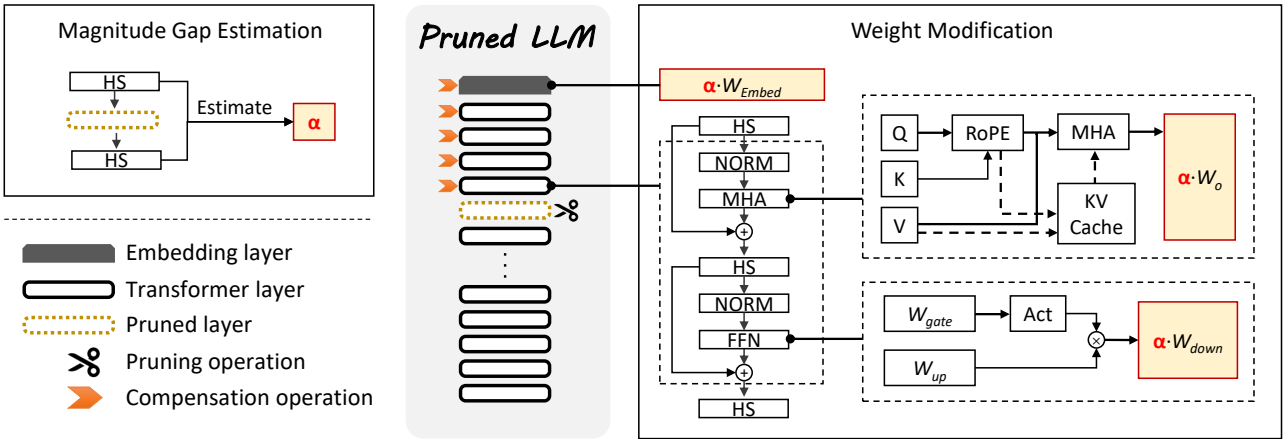

Figure 4: Magnitude compensation applied to LLM. HS: hidden states; MHA: multi-head attention; FFN: feed-forward network; W_{Embed}: embedding weight; NORM: normalization layer; RoPE: rotary position embedding; $W_{gate}, W_{up}, W_{down}$: FFN gate, up and down projection weight, respectively; W_O: MHA output projection weight; Act: activation function.

Perplexity (PPL) (Song et al. 2024; Kim et al. 2024): Lower PPL indicates the fluency of LLM text generation. Redundant blocks contribute less to the model performance, and their removal leads to a smaller increase in PPL. The PPL importance score I^n_{PPL} of the n-th block is defined as:

$$I^{\ell}_{PPL} = \exp\left\{-\frac{1}{ST}\sum_{s=1}^{S}\sum_{t=1}^{T}\log\, p_{\theta^{\ell}}(x_t^{(s)}|x_{<t}^{(s)})\right\}, \quad (6)$$

where θ^{ℓ} denotes the model without its ℓ-th block, $s = 1,\ldots,S$ are the indices for sequences, and $t = 1,\ldots,T$ are the indices for tokens in calibration set D.

Taylor+ (Kim et al. 2024): Taylor metrics estimate the significance of a weight parameter by assessing the error caused by its removal. For a given calibration dataset D, this can be expressed as the alteration in the training loss $\mathcal{L}$:

$$|\mathcal{L}(W^{k,\ell}_{i,j}; D) - \mathcal{L}(W^{k,\ell}_{i,j} = 0; D)| \approx \left|\frac{\partial \mathcal{L}(D)}{\partial W^{k,\ell}_{i,j}} W^{k,\ell}_{i,j}\right|,$$

where second-order derivatives are omitted. The Taylor+ block importance score I^{ℓ}_{Taylor} can be defined as:

$$I^{\ell}_{Taylor} = \sum_k \sum_i \sum_j \left|\frac{\partial \mathcal{L}(D)}{\partial W^{k,\ell}_{i,j}} W^{k,\ell}_{i,j}\right|, \quad (7)$$

where $W^{k,\ell}$ is the linear weight matrix of operation type k in the ℓ-th Transformer block, and $W^{k,\ell}_{i,j}$ is its element. The Taylor+ method builds upon the Taylor metric by preserving the first four and the last two Transformer blocks, as their removal leads to severe performance drops. A lower I^{ℓ}_{Taylor} score indicates a block that is less important and therefore more suitable for pruning.

Magnitude+ (Mag+) (Kim et al. 2024): Mag+ incorporates a heuristic rule based on the Magnitude metric (referencing Li et al., 2017b, which assumes weights with smaller norms are less informative) and preserves the first four and the last two blocks of the model. The underlying Mag+ importance score can be calculated as:

$$I^{\ell}_{Magnitude} = \sum_k \sum_i \sum_j |W^{k,\ell}_{i,j}|, \quad (8)$$

where $W^{k,\ell}_{i,j}$ is an element of the linear weight matrix of operation type k in the ℓ-th Transformer block. A lower $I^{\ell}_{Magnitude}$ score indicates a block that is less important and therefore more suitable for pruning.

Magnitude Compensation

Magnitude Gap Estimation. Once the layer importance is determined by a certain pruning metric, the least-important layer is dropped as Equation 2. As motivated earlier, removing any layer introduces a magnitude gap that is independent of the pruning metric. To compensate for this gap, we estimate the magnitude gap on a small calibration set before pruning. The goal is to obtain an optimal magnitude compensation scalar factor α. Assuming that layer ℓ is to be removed, α is defined as:

$$\alpha = \mathbb{E}_{(X^{(\ell)}, X^{(\ell+1)})\in\mathcal{D}} \frac{1}{C}\sum_{k=1}^{C} \frac{\|X^{(\ell+1)}_{:,k}\|_1}{\|X^{(\ell)}_{:,k}\|_1}, \quad (9)$$

where $X^{(\ell)}$ and $X^{(\ell+1)}$ are the hidden-states at the input and output of layer ℓ, respectively, collected over the calibration samples. After α is estimated, we perform layer pruning and scale the input to layer $\ell+1$ with α as compensation. Formally, the forward propagation with compensation becomes:

$$X^{(\ell+1)} = \alpha X^{(\ell)} + f(\alpha X^{(\ell)}, \theta^{(\ell+1)}). \quad (10)$$

Weight Modification. Equation 10 scales hidden states by the scalar α via an element-wise multiplication during inference time, incurring online overhead. We eliminate this cost by fusing α directly into the weights of model layers that precede the pruned layer. As illustrated in Figure 4, once α is estimated, it is fused into the model weights in three steps.

LLaMA-2-7B						LLaMA-3-8B					
Sparsity	**Metric**	**WikiText-2**	**C4**	**PTB**	**Average**	**Sparsity**	**Metric**	**WikiText-2**	**C4**	**PTB**	**Average**
-	Dense	5.47	6.97	22.51	11.65	-	Dense	6.14	8.88	10.59	8.54
7/32	PPL	9.81	12.36	48.53	23.57	5/32	PPL	12.37	15.28	18.91	15.52
	+PRUNE&COMP	**8.57**	**10.55**	**40.51**	**19.88**		+PRUNE&COMP	**9.65**	**13.20**	**16.02**	**12.96**
	CosSim(CL)	18.45	20.99	62.18	33.87		CosSim(CL)	21.14	24.13	37.41	27.56
	+PRUNE&COMP	**13.78**	**15.31**	**49.40**	**26.16**		+PRUNE&COMP	**16.90**	**18.57**	**21.64**	**19.04**
	Mag+	49.39	34.65	184.78	89.61		Mag+	37.57	34.99	60.80	44.45
	+PRUNE&COMP	**11.73**	**13.28**	**59.60**	**28.20**		+PRUNE&COMP	**13.60**	**18.24**	**22.52**	**18.12**
	Taylor+	18.45	20.99	63.02	34.15		Taylor+	602.96	388.41	546.98	512.78
	+PRUNE&COMP	**10.61**	**12.10**	**51.02**	**24.58**		+PRUNE&COMP	**12.78**	**16.20**	**20.03**	**16.34**
	CosSim(BI)	18.45	20.99	62.18	33.87		CosSim(BI)	27.33	27.06	31.81	28.73
	+PRUNE&COMP	**11.45**	**12.96**	**42.29**	**22.23**		+PRUNE&COMP	**11.87**	**14.87**	**16.93**	**14.56**
9/32	PPL	14.91	17.03	67.73	33.22	7/32	PPL	15.08	17.57	22.09	18.2
	+PRUNE&COMP	**10.23**	**12.08**	**50.96**	**24.42**		+PRUNE&COMP	**12.40**	**15.98**	**20.27**	**16.22**
	CosSim(CL)	35.68	36.10	96.52	56.10		CosSim(CL)	2287.73	1491.37	4738.81	2839.30
	+PRUNE&COMP	**19.37**	**20.13**	**58.20**	**32.57**		+PRUNE&COMP	**204.06**	**231.6**	**256.53**	**230.73**
	Mag+	362.15	48.79	273.07	228.00		Mag+	40.70	36.95	44.85	40.83
	+PRUNE&COMP	**19.09**	**18.88**	**95.98**	**44.65**		+PRUNE&COMP	**33.33**	**35.02**	**42.93**	**37.09**
	Taylor+	35.68	36.10	96.52	56.10		Taylor+	2287.86	1491.38	4741.9	2840.38
	+PRUNE&COMP	**13.80**	**14.52**	**69.04**	**32.45**		+PRUNE&COMP	**21.10**	**22.05**	**32.47**	**25.21**
	CosSim(BI)	35.68	36.10	96.52	56.10		CosSim(BI)	57.76	50.13	67.39	58.43
	+PRUNE&COMP	**18.53**	**17.99**	**60.38**	**32.30**		+PRUNE&COMP	**28.43**	**24.48**	**28.57**	**27.16**

Table 1: Performance comparison on perplexity (PPL) benchmark. Sparsity is indicated by layers pruned/total layers.

Step 1: Modifying embedding layer. The weight of token-embedding layer W_{embed} is updated:

$$W_{embed} \leftarrow \alpha W_{embed}. \tag{11}$$

Step 2: Modifying MHA output projections. For every layer with index k∈[1, ℓ-1], the MHA output-projection matrix W_o is updated:

$$W_o^{(k)} \leftarrow \alpha W_o^{(k)}. \tag{12}$$

Step 3: Modifying MLP down projections. Likewise, the down-projection weight W_{Down} of every MLP layer with index k∈[1, ℓ-1] is scaled:

$$W_{down}^{(k)} \leftarrow \alpha W_{down}^{(k)}. \tag{13}$$

We explain the rationale of the weight modification step-by-step. With Step 1, the embedding layer's output hidden states are scaled by α. Subsequent Transformer layers consist of a Multi-Head Attention (MHA), a Multi-Layer Perceptron (MLP), and a normalization layer (NORM). The first Transformer layer receives the scaled hidden states from the embedding layer. The hidden states are then (i) copied as a residual branch, (ii) normalized, (iii) fed to MHA, and (iv) added to the residual. For a normalization layer that is scale-invariant, i.e., NORM(X)=NORM(αX), we re-introduce the factor α and multiply the output projection weights of MHA by α, which effectively scales the MHA output by α before the residual addition. The same logic applies to the MLP block. The scaled MHA output is copied as a residual, normalized, and passed through the MLP. Owing again to normalization's scale invariance, we multiply the down-projection weights of the MLP by α, preserving the magnitude. We then apply this procedure to every layer preceding the pruned layer ℓ. With these modifications, the model architecture remains unchanged except for the removed layers, incurring no additional online cost at inference time.

Iterative Pruning with Magnitude Compensation (PRUNE&COMP)

Algorithm 1: The proposed PRUNE&COMP algorithm.

$M \leftarrow$ *original model*
$C \leftarrow$ *calibration dataset*
$N \leftarrow$ *# blocks of* M
$n \leftarrow$ *# blocks to remove*
for $i = 0$ **to** $n - 1$ **do**
  $idx \leftarrow \text{Metric}(M, C)$
  $M \leftarrow \text{Prune}(M, idx)$
  $M \leftarrow \text{Comp}(M, idx)$
end for

Furthermore, we combine iterative pruning with magnitude compensation. PRUNE&COMP iteratively removes layer from an original model M until a target number of n layers has been eliminated, as summarized in Algorithm 1 . In each iteration, the algorithm performs the following three steps:

- $Metric(M, C)$: Given model M, selects the most redundant layer for removal with a chosen pruning metric over a calibration dataset C, and returns its index idx.
- $Prune(M, idx)$: Removes the block at index idx from the model M and return the pruned model.
- $Comp(M, idx)$: Compensates the model M for the magnitude gap that arises in the hidden states due to layer removal and returns the compensated model.

By actively compensating for this gap through training-free magnitude gap estimation and weight modifications, PRUNE&COMP ensures that the pruned model maintains a robust internal representation, which in turn enhances the accuracy of subsequent layer importance estimations and ultimately preserves performance. This iterative cycle of pruning and compensation enables the algorithm to effectively

Model	Sparsity	Metric	ARC-c	ARC-e	BoolQ	CoPa	HeSw	PIQA	Race-h	WG	WSC	Average	RP
LLaMA-3-8B	**0/32**	Dense	53.41	77.78	81.28	89.00	79.16	80.85	40.19	72.85	86.45	73.44	100.00
	5/32 (13.58%)	PPL	32.76	61.36	**56.42**	75.00	61.77	75.52	32.25	54.22	65.20	57.17	77.84
		+PRUNE&COMP	**40.87**	**66.58**	56.27	**85.00**	**67.44**	**76.22**	**33.97**	**62.59**	**73.26**	**62.47**	**85.06**
		CosSim(CL)	47.35	66.20	73.52	84.00	71.10	74.27	36.65	71.03	76.56	66.74	90.88
		+PRUNE&COMP	**48.63**	**69.99**	**74.07**	**85.00**	**72.63**	**75.95**	**37.51**	**72.61**	**79.12**	**68.39**	**93.12**
		Mag+	29.95	56.36	53.21	73.00	40.35	69.64	27.18	52.80	60.44	51.44	70.04
		+PRUNE&COMP	**32.76**	**57.87**	**58.96**	**82.00**	**59.85**	**73.23**	**30.72**	**55.25**	**66.30**	**57.44**	**78.21**
		Taylor+	33.53	45.58	55.00	55.00	35.86	59.52	24.31	60.85	65.57	48.36	65.85
		+PRUNE&COMP	**46.16**	**70.24**	**69.36**	**81.00**	**69.67**	**73.83**	**37.99**	**70.56**	**79.85**	**66.52**	**90.57**
		CosSim(BI)	45.56	63.51	**73.12**	79.00	70.13	74.92	36.94	71.19	75.09	65.50	89.18
		+PRUNE&COMP	**46.50**	**70.54**	71.31	**84.00**	**72.43**	**76.01**	**37.99**	**73.32**	**83.88**	**68.44**	**93.19**
	7/32 (19.01%)	PPL	32.76	58.84	45.38	75.00	**59.22**	**73.56**	**30.72**	53.83	**67.77**	55.23	75.20
		+PRUNE&COMP	**33.53**	**60.48**	**47.52**	**76.00**	59.03	**73.56**	**30.72**	**54.54**	67.40	**55.86**	**76.07**
		CosSim(CL)	28.92	39.56	38.07	60.00	33.26	59.47	24.02	55.56	59.71	44.29	60.30
		+PRUNE&COMP	**32.76**	**45.62**	**52.20**	**66.00**	**43.41**	**63.55**	**27.66**	**57.85**	**62.64**	**50.19**	**68.34**
		Mag+	25.60	46.04	56.18	70.00	43.36	64.91	27.46	53.43	55.31	49.14	66.92
		+PRUNE&COMP	**30.63**	**49.37**	**58.10**	**76.00**	**51.31**	**68.82**	**28.71**	**55.01**	**60.81**	**53.20**	**72.43**
		Taylor+	29.01	39.56	38.00	60.00	33.24	59.30	24.02	55.49	59.71	44.26	60.27
		+PRUNE&COMP	**42.66**	**67.51**	**67.61**	**78.00**	**65.84**	**72.36**	**37.22**	**70.01**	**77.66**	**64.32**	**87.58**
		CosSim(BI)	**42.41**	56.65	65.26	75.00	64.70	70.89	34.16	**71.19**	73.63	61.54	83.80
		+PRUNE&COMP	41.55	**62.37**	**72.78**	**80.00**	**65.44**	**71.38**	**36.65**	71.11	**78.02**	**64.37**	**87.64**
Qwen3-8B	**0/32**	Dense	56.57	80.93	86.57	85.00	74.93	77.80	40.96	67.80	83.15	72.63	100.00
	5/36 (11.78%)	PPL	**46.93**	**74.41**	69.63	**82.00**	**76.06**	**77.20**	**38.37**	**61.25**	**75.46**	**66.81**	**91.98**
		+PRUNE&COMP	46.84	74.33	**75.17**	78.00	61.39	76.44	37.80	58.88	74.36	64.80	89.22
		CosSim(CL)	42.41	61.15	77.98	73.00	58.80	67.30	33.78	65.27	72.16	61.32	84.42
		+PRUNE&COMP	**43.17**	**65.57**	**86.30**	**77.00**	**60.82**	**69.15**	**37.03**	**67.40**	**77.29**	**64.86**	**89.29**
		Mag+	42.75	70.03	**77.00**	77.00	**63.86**	**75.90**	36.46	58.56	73.26	63.87	87.93
		+PRUNE&COMP	**45.31**	**75.51**	72.75	**82.00**	61.82	74.27	**39.52**	**60.54**	**74.73**	**65.16**	**89.71**
		Taylor+	40.02	63.38	55.90	**73.00**	62.05	68.99	**37.89**	59.98	71.79	59.22	81.53
		+PRUNE&COMP	**43.09**	**66.08**	**82.91**	72.00	**62.57**	**69.75**	37.51	**62.83**	**78.39**	**63.90**	**87.98**
		CosSim(BI)	46.42	73.74	77.40	80.00	**64.54**	76.82	**37.89**	62.75	76.19	66.19	91.13
		+PRUNE&COMP	**47.27**	**74.07**	**81.56**	**82.00**	63.54	**77.09**	37.80	**63.06**	**79.12**	**67.28**	**92.63**
	7/36 (16.49%)	PPL	41.13	68.64	67.77	74.00	**60.44**	74.59	35.02	55.64	69.23	60.72	83.59
		+PRUNE&COMP	**43.60**	**72.26**	**68.23**	**77.00**	58.64	**75.03**	**36.84**	**56.99**	**72.16**	**62.31**	**85.78**
		CosSim(CL)	33.87	48.57	73.55	69.00	50.97	61.86	31.96	59.98	67.03	55.20	76.00
		+PRUNE&COMP	**35.67**	**51.52**	**84.59**	**72.00**	**54.56**	**64.53**	**34.45**	**62.59**	**74.36**	**59.36**	**81.73**
		Mag+	**40.10**	**68.27**	55.60	71.00	**58.9**	**73.12**	33.59	56.75	69.23	58.51	80.56
		+PRUNE&COMP	37.54	63.55	**76.33**	**75.00**	55.80	71.22	**35.89**	**59.04**	**71.06**	**60.60**	**83.44**
		Taylor+	35.75	49.62	45.17	**67.00**	53.29	64.53	34.64	57.54	64.47	52.45	72.20
		+PRUNE&COMP	**36.60**	**54.50**	**85.50**	**67.00**	**56.31**	**65.78**	**35.50**	**62.51**	**72.53**	**59.58**	**82.03**
		CosSim(BI)	**42.49**	69.07	68.35	**80.00**	**61.48**	**75.24**	**35.50**	54.38	67.77	61.59	84.79
		+PRUNE&COMP	40.70	**69.44**	**74.07**	76.00	58.86	74.70	34.93	**57.38**	**68.86**	**61.66**	**84.89**

Table 2: Performance comparison on question answering (QA) benchmark. Sparsity is indicated by layers pruned/total layers (compression rate). RP denotes the relative performance (%).

reduce model depth while minimizing performance degradation, resulting in a more streamlined and efficient LLM that avoids incurring online inference overhead.

Experiments

Experimental Setup and Details

Models. We evaluate PRUNE&COMP on open-sourced LLMs including LLaMA-2-7B/13B (Touvron et al. 2023), LLaMA-3-8B (Dubey et al. 2024), Qwen3-8B (Team 2025).

Baseline. We compare with the prevalent layer-wise pruning metrics in one-shot manner, including cosine similarity-based metrics (Men et al. 2024; Chen et al. 2024a; Gromov et al. 2024), perplexity (PPL)-based metrics (Song et al. 2024; Kim et al. 2024), Taylor+ (Kim et al. 2024), and Magnitude+ (Kim et al. 2024), as presented in Methods. The pruning settings are provided in the Appendix. The calibration process uses 128 randomly selected sequences, each comprising 2048 tokens, from the WikiText-2 dataset to determine the layer to prune and initialize the magnitude compensation factor. The computational cost of solving for the compensation factor is approximately one minute per pruned layer on the 7B model, while the time required for weight modification is negligible. All experiments were performed on a 24GB NVIDIA V100 GPU.

Evaluation. Three benchmarks are used: perplexity (PPL) including WikiText2 (Merity et al. 2016), C4 (Raffel et al. 2020), and PTB (Marcus, Santorini, and Marcinkiewicz 1993); massive multitask language understanding (MMLU) (Hendrycks et al. 2020); commonsense question answering (QA) including ARC-Challenge (ARC-c), ARC-Easy (ARC-e) (Clark et al. 2018), BoolQ (Clark et al. 2019), HellaSwag (HeSw) (Zellers et al. 2019), PIQA (Bisk et al. 2020), WinoGrande (WG) (Sakaguchi et al. 2020), WSC273 (WSC) (Levesque, Davis, and Morgenstern 2012), Race-high (Race-h) (Lai et al. 2017) and CoPA (Sarlin et al. 2020).

Results on PPL Benchmark

Table 1 presents an analysis of the perplexity benchmark for LLaMA-2-7B and LLaMA-3-8B across diverse pruning configurations. We report the average performance on WikiText-2, C4, and PTB datasets. Results of more models, including LLaMA-2-13B, are provided in the Appendix.

For the LLaMA-2-7B, the plug-and-play PRUNE&COMP consistently improves baselines across all pruning metrics and datasets. For example, when pruning 7 out of 32 layers with CosSim(BI) metric, combining with PRUNE&COMP drops the average PPL from 23.57 to 19.88, while with Mag+ metric PRUNE&COMP improves PPL dramatically from 89.61 to 28.20. This trend of substantial PPL reduction via PRUNE&COMP persists at the higher sparsity, demonstrating its effectiveness in mitigating layer pruning-induced performance degradation. Likewise, LLaMA-3-8B consistently reaps large gains from PRUNE&COMP, and combining with every baseline achieves lower perplexity across the metrics. The most striking example is Taylor+ with 5 of 32 layers pruned: its average PPL collapses from an extreme 512.78 to a healthy 16.34, when integrating with PRUNE&COMP. The benefit of PRUNE&COMP widens when 7 out of 32 layers are pruned. Taylor+ metric, combining with PRUNE&COMP plunges from 2840.38 to 25.21, and CosSim(CL) metric, combining with PRUNE&COMP likewise tumbles from 2839.30 to 230.73.

Results on QA Benchmark

Table 2 presents a comprehensive analysis on question answering (QA) benchmark of LLaMA-3-8B and Qwen3-8B across diverse pruning configurations. Results of more models, including LLaMA-2-13B, refer to the Appendix.

For LLaMA-3-8B model, the combination of PRUNE&COMP consistently enhances the pruned model's performance across all pruning metrics when compared to their baselines, demonstrating its effectiveness. For instance, when pruning 5 out of 32 layers with the Taylor+ metric, the related performance significantly increases from 65.85% to 90.57%, leading the baseline with 24.72%. Similarly, with 7 out of 32 layers pruned, PRUNE&COMP raises the related performance from 60.27 to 87.58 using the Taylor+ metric, significantly higher than the baseline. For Qwen3-8B, a similar positive trend is observed. By integrating PRUNE&COMP, the model's relative performance improved across all pruning metrics. For example, when pruning 5 out of 32 layers with CosSim(CL) metric, PRUNE&COMP boosts relative performance from 84.42% to 89.29%, surpassing the baseline by 4.87%.

Results on MMLU Benchmark

Further analysis of the massive multitask language understanding (MMLU) benchmark is summarized in Table 3. For most sub-tasks, the conventional pruning methods combined with PRUNE&COMP yield superior performance, demonstrating their robust capability to improve model performance across diverse knowledge domains.

For example, when pruning 5 out of 32 layers of LLaMA-3-8B with the Taylor+ metric, PRUNE&COMP significantly boosts the weighted accuracy from 38.64% to 48.42%, leading the baseline by 9.78%. Likewise, for the CosSim(CL) metric, the performance is improved from 62.40% to 63.55%. These results indicate that PRUNE&COMP maintains superior performance across multiple pruning metrics.

Metric	STEM	Human.	Soc. Sci.	Others	Weighted Acc.
PPL	28.76	24.36	27.10	28.22	26.80
+PRUNE&COMP	**30.72**	**25.59**	**32.17**	**30.44**	**29.25**
CosSim(CL)	53.47	56.08	74.58	68.32	62.40
+PRUNE&COMP	**54.31**	**56.98**	**75.04**	**70.76**	**63.55**
Mag+	27.24	23.53	24.76	**27.64**	25.54
+PRUNE&COMP	**28.10**	**24.14**	**31.04**	25.91	**26.91**
Taylor+	31.41	37.4	44.65	41.39	38.64
+PRUNE&COMP	**39.36**	**43.97**	**57.26**	**54.90**	**48.42**
CosSim(BI)	46.92	**53.92**	65.65	65.42	57.64
+PRUNE&COMP	**49.93**	52.09	**69.45**	**67.46**	**58.98**

Table 3: Performance comparison on MMLU benchmark. 5 layers of LLaMA-3-8B are pruned with CosSim(BI) metric.

Method	WikiText-2	C4	PTB	Average
Naive one-shot pruning	57.76	50.13	67.39	58.43
+IterPrune	36.91	32.2	46.00	38.37
+MagComp	35.38	33.71	43.08	37.39
+MagComp +IterPrune	**28.43**	**24.48**	**28.57**	**27.16**

Table 4: Effectiveness of the proposed PRUNE&COMP. 7 layers of LLaMA-3-8B are pruned with CosSim(BI) metric.

Ablation Studies

Table 4 investigates the individual contribution of PRUNE&COMP, including iterative pruning (+IterPrune) and magnitude compensation (+MagComp), on the perplexity benchmark. Both iterative pruning and magnitude compression individually yield significant improvements over naive one-shot pruning across WikiText-2, C4, and PTB datasets. Specifically, with naive one-shot pruning, results in an average perplexity of 58.43, iterative pruning reduces this to 38.37, while magnitude compression achieves an average of 37.39. Crucially, the combination of both achieves the best average perplexity of 27.16, demonstrating a strong synergistic effect where their combined application substantially outperforms individual components. We also provide ablation on QA benchmarks in the Appendix.

Conclusion

We present PRUNE&COMP, a training-free plug-and-play iterative layer pruning scheme that mitigates the magnitude gap from layer removal in LLMs via magnitude compensation. By integrating this compensation with iterative pruning, which involves estimating the gap induced by layer pruning and closing it through offline weight rescaling, our method avoids inference overhead while addressing hidden states discrepancy-driven performance degradation. PRUNE&COMP consistently enhances existing layer pruning metrics, validating its effectiveness for practical LLM compression. This work facilitates LLM compression in resource-constrained settings and marks a step toward developing training-free layer pruning schemes.

Acknowledgments

This work was jointly supported by the National Key R&D Programs of China (2022YFB4701400/4701402) and SSTIC Grants (KJZD20230923115106012, KJZD20230923114916032, GJHZ20240218113604008).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

An, Y.; Zhao, X.; Yu, T.; Tang, M.; and Wang, J. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 10865–10873.

Ashkboos, S.; Croci, M. L.; Nascimento, M. G. d.; Hoefler, T.; and Hensman, J. 2024. SliceGPT: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, 7432–7439.

Chen, X.; Hu, Y.; Zhang, J.; Wang, Y.; Li, C.; and Chen, H. 2024a. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

Chen, X.; Wang, Y.; Li, Y.; Ling, X.; Li, M.; Liu, R.; Ouyang, M.; Zhao, K.; Guan, T.; and He, Y. 2024b. Low Bit-Width Zero-Shot Quantization With Soft Feature-Infused Hints for IoT Systems. *IEEE Internet of Things Journal*.

Chen, X.; Yan, R.; Cheng, J.; Wang, Y.; Fu, Y.; Chen, Y.; Guan, T.; and He, Y. 2023. ADEQ: Adaptive diversity enhancement for zero-shot quantization. In *International Conference on Neural Information Processing*, 53–64. Springer.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The LLaMA 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gromov, A.; Tirumala, K.; Shapourian, H.; Glorioso, P.; and Roberts, D. 2024. The Unreasonable Ineffectiveness of the Deeper Layers. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hu, Y.; Zhang, J.; Zhao, Z.; Zhao, C.; Chen, X.; Li, C.; and Chen, H. 2024. SP3: Enhancing Structured Pruning via PCA Projection. In *Findings of the Association for Computational Linguistics ACL 2024*, 3150–3170.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Kim, B.-K.; Kim, G.; Kim, T.-H.; Castells, T.; Choi, S.; Shin, J.; and Song, H.-K. 2024. Shortened LLaMA: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Levesque, H.; Davis, E.; and Morgenstern, L. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Li, C.; Chen, X.; Wang, J.; Zhao, K.; and Chen, J. 2025. Task-Specific Zero-shot Quantization-Aware Training for Object Detection. *arXiv preprint arXiv:2507.16782*.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36: 21702–21720.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330.

Men, X.; Xu, M.; Zhang, Q.; Wang, B.; Lin, H.; Lu, Y.; Han, X.; and Chen, W. 2024. ShortGPT: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Muralidharan, S.; Turuvekere Sreenivas, S.; Joshi, R.; Chochowski, M.; Patwary, M.; Shoeybi, M.; Catanzaro, B.; Kautz, J.; and Molchanov, P. 2024. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37: 41076–41102.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Sakaguchi, K.; Le Bras, R.; Bhagavatula, C.; and Choi, Y. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8732–8740.

Sarah, A.; Sridhar, S. N.; Szankin, M.; and Sundaresan, S. 2024. LLaMA-NAS: Efficient Neural Architecture Search for Large Language Models. *arXiv preprint arXiv:2405.18377*.

Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4938–4947.

Song, J.; Oh, K.; Kim, T.; Kim, H.; Kim, Y.; and Kim, J.-J. 2024. SLEB: Streamlining LLMs through Redundancy Verification and Elimination of Transformer Blocks. *arXiv preprint arXiv:2402.09025*.

Sreenivas, S. T.; Muralidharan, S.; Joshi, R.; Chochowski, M.; Mahabaleshwarkar, A. S.; Shen, G.; Zeng, J.; Chen, Z.; Suhara, Y.; Diao, S.; et al. 2024. LLM pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Sun, Y.; Liu, R.; Bai, H.; Bao, H.; Zhao, K.; Li, Y.; Hu, J.; Yu, X.; Hou, L.; Yuan, C.; et al. 2024. FlatQuant: Flatness matters for LLM quantization. *arXiv preprint arXiv:2410.09426*.

Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; et al. 2025. Kimi k1. 5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*.

Team, Q. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

van der Ouderaa, T. F.; Nagel, M.; Van Baalen, M.; Asano, Y. M.; and Blankevoort, T. 2023. The LLM surgeon. *arXiv preprint arXiv:2312.17244*.

Xia, M.; Gao, T.; Zeng, Z.; and Chen, D. 2023. Sheared LLaMA: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.