A Simple Linear Patch Revives Layer-Pruned Large Language Models

Anonymous ACL submission

Abstract

Layer pruning has become a popular technique for compressing large language models (LLMs) due to its simplicity. However, existing layer pruning methods often suffer from significant performance drops. We identify that *this degradation stems from the mismatch of activation magnitudes across layers and tokens at the pruning interface*. To address this, we propose LINEARPATCH, a simple plug-and-play technique to revive the layer-pruned LLMs. Our method introduces Hadamard transformation to suppress massive outliers in particular tokens, and channel-wise scaling to align the activation magnitudes. These operations can be ingeniously fused into a real symmetric matrix using the spectral theorem, i.e., the proposed LINEARPATCH at the pruning interface with negligible inference overhead. Our experiments demonstrate that LINEARPATCH retains up to **94.15%** performance of the original model when pruning 5 layers of LLaMA-3-8B on the question answering benchmark, surpassing existing state-of-the-art methods by **4%**. Additionally, with the proposed offline knowledge distillation using only 5K samples, LINEARPATCH can be further boosted to **95.16%** within 30 minutes on a single computing card. Code will be released upon acceptance.

1 Introduction

Recent large language models (LLMs) have achieved remarkable success towards the artificial general intelligence (Achiam et al., 2023; Jiang et al., 2023; Yang et al., 2024a; Dubey et al., 2024; Team et al., 2025; Liu et al., 2024a; Guo et al., 2025). However, the large size of LLMs also brings significant computational and memory overhead for the deployment. Various model compression methods are thereon proposed, including quantization (Xiao et al., 2023a; Ashkboos et al., 2024b; Sun et al., 2024b) and pruning (Ashkboos et al., 2024a; van der Ouderaa et al., 2023; Xia et al., 2023; Sarah et al., 2024; Hu et al., 2024).

Among these methods, layer pruning is a popular solution as it can be readily applied without hardware or low-level dependencies (Song et al., 2024; Kim et al., 2024; Chen et al., 2024a; Men et al., 2024; Gromov et al., 2024). This is different from unstructured pruning (Han et al., 2015; Frantar and Alistarh, 2023; Sun et al., 2023) which cannot be efficiently accelerated due to the irregular memory access. The structured sparsity (Ashkboos et al., 2024a; van der Ouderaa et al., 2023) or N:M sparsity (Frantar and Alistarh, 2023; Sun et al., 2023; Zhang et al., 2024) also rely on hacking the model architectures or low level kernels. However, despite the simplicity of layer pruning, a critical challenge is the sharp drop of performance.

In this work, we discover a new phenomenon that explains the degradation by layer pruning, i.e., *the mismatch of activation magnitudes across layers and tokens at the pruning interface*. Specifically, when layers are pruned in LLMs, the activations from the remaining layers often exhibit different scales. The magnitude of activations in the layer preceding the pruning interface may not match those in the following layer after pruning. The mismatch is exacerbated by the presence of massive outliers in the activations of special tokens (e.g., `[BOS]` or delimiter tokens), which are common in LLMs (Liu et al., 2024b; Sun et al., 2024a). As a result, the pruned LLMs may suffer from efficient forward propagation as before, which ultimately leads to the drop in performance.

To this end, we propose LINEARPATCH, a plug-and-play technique to compensate the mismatch of activation magnitudes. The proposed method can be orthogonally integrated with various pruning metrics. Specifically, LINEARPATCH first applies Hadamard transformation to suppress the massive outliers from special tokens (e.g., `[BOS]` or delimiter tokens), a well-known issue in LLMs (Liu et al., 2024b; Sun et al., 2024a). A channel-wise scaling parameter is then introduced to bridge the gap of

activation magnitudes. *By leveraging the spectrum theory, both the Hadamard matrices and diagonalized channel-wise scaling can be integrated into a real symmetric matrix, i.e., the proposed* LINEARPATCH *to be inserted at the pruning interface.* The proposed approach introduces negligible inference overhead, and is shown to effectively align the activation magnitudes. To further enhance the pruned LLMs, we explore the potential of offline knowledge distillation. Specifically, we fine-tune only the linear patch introduced by LINEARPATCH while freezing all other parameters. This efficient training process requires only 5K samples and can be completed within 30 minutes on a single computing card for the LLaMA-2-7B model.

Our empirical results demonstrate the effectiveness of LINEARPATCH across various LLMs and tasks. For instance, LINEARPATCH achieves a retained performance ratio of up to **94.15%** when pruning 5 layers for LLaMA-3-8B on the question answering benchmark, significantly higher than existing state-of-the-art (SOTA) methods such as LLM-Streamline (90.84%). Moreover, with efficient knowledge distillation, LINEARPATCH can further boost the retained performance to **95.16%**. These results highlight the potential of LINEARPATCH as a powerful solution for reviving layer-pruned LLMs with minimal overhead and significant performance gains.

2 Related work

Recently, layer pruning has emerged as a promising approach for compressing Large Language Models (LLMs). Unlike width pruning that often results in irregular network structures, layer pruning removes entire Transformer layers, including attention and MLP modules, making it easier to deploy. Recent studies, such as LaCo (Yang et al., 2024b), ShortGPT (Men et al., 2024), UIDL (Gromov et al., 2024), SLEB (Song et al., 2024), Shortened LLaMa (Kim et al., 2024), and LLM-Streamline (Chen et al., 2024a), have demonstrated the effectiveness of layer pruning in compressing LLMs.

LoCo (Yang et al., 2024b) merges rear model layers into a prior layer using reserving differences while seeking common (RDSC) strategy, with cosine similarity of the model's output as the guiding metric. ShortGPT (Men et al., 2024) employs cosine similarity between the inputs and outputs of each layer to assess layer importance, subsequently removing the least critical layers. UIDL (Gromov et al., 2024) introduces an angular distance metric to evaluate and remove consecutive layers, followed by QLoRA fine-tuning to mitigate pruning-induced damage. SLEB (Song et al., 2024) iteratively identifies and eliminates redundant layers based on perplexity degradation on a calibration dataset. Shortened LLaMa (Kim et al., 2024) explores Taylor-based metrics and perplexity degradation as pruning criteria, restoring pruned model performance through LoRA fine-tuning. LLM-Streamline (Chen et al., 2024a) identifies the least important consecutive layers using cosine similarity and replaces them with a lightweight network, claiming that fine-tuning this network with MSE loss outperforms LoRA fine-tuning.

Despite the demonstrated effectiveness of these layer pruning methods, they all overlook the significant magnitude mismatch that occurs after pruning, which we find detrimental to model performance. LINEARPATCH addresses this issue by introducing a simple yet effective magnitude alignment patch. While LLM-Streamline also inserts a lightweight network at the pruning site, their network is randomly initialized and fails to address magnitude alignment. In contrast, LinearPatch achieves superior performance under both training-free and post-training conditions compared to concurrent layer pruning methods.

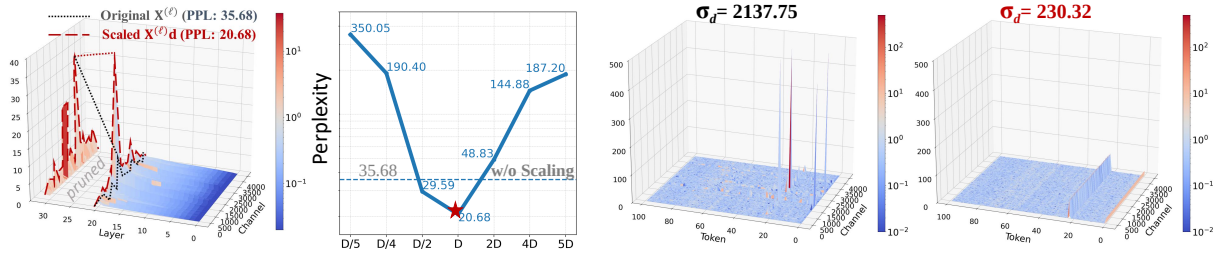
3 Method

We introduce LINEARPATCH, a plug-and-play method to enhance the layer-pruned LLMs. We find that channel magnitude matters both across different layers (Section 3.2) and tokens (Section 3.3). In Section 3.4, we introduce that all these issues can be resolved by a linear patch, which can be efficiently fine-tuned for further improvement.

3.1 Preliminaries on LLM Layer Pruning

Layer pruning has emerged as a promising technique for compressing the redundant layers in large language model. LLMs primarily adopt the Transformer architecture, which consists of a series of Transformer decoder layers and each layer adopts a residual structure. We denote the ℓ-th Transformer layer as $f(\mathbf{X}^{(\ell)}; \theta^{(\ell)})$, with $\mathbf{X}^{(\ell)}$ and $\theta^{(\ell)}$ representing its input activations and associated parameters, respectively. Due to the prevalent pre-norm architecture of LLMs, the input to the $(\ell+1)$-th layer $\mathbf{X}^{(\ell+1)}$ can thus be obtained by

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



(a) Layer-wise Channel Magnitude (b) Channel Magnitude Scaling (c) Token-wise Channel Magnitude (d) Hadamard Trans. Magnitude

Figure 1: Visualization of the identified layer-wise channel mismatch and token-wise channel mismatch in layer-pruned LLMs. (a) and (b) identify the layer-wise channel mismatch at the pruning interface and the validity of channel scaling. (c) and (d) identify the token-wise channel mismatch and the validity of Hadamard transformation.

Cosine Similarity for Layer Pruning. We choose the prevalent cosine similarity as the pruning metric, despite that the proposed approach is agnostic to the pruning metric. Recent efforts find cosine similarity particularly effective in identifying redundant Transformer layers (Men et al., 2024; Gromov et al., 2024). For example, LLM-Streamline (Chen et al., 2024a) identifies and removes n contiguous layers with the highest cosine similarities among their input activations. The optimal layer index ℓ^* is thus determined by:

$$\ell^* = \arg\max_{\ell} \mathbb{E}_{(\mathbf{X}_i^{(\ell)}, \mathbf{X}_i^{(\ell+n)}) \in \mathcal{D}} \frac{\mathbf{X}_i^{(\ell)} \cdot \mathbf{X}_i^{(\ell+n)}}{\|\mathbf{X}_i^{(\ell)}\| \|\mathbf{X}_i^{(\ell+n)}\|}, \quad (2)$$

where $\mathcal{D}$ denotes the input activations of layers obtained with calibration samples, and $\mathbf{X}_i^{(\ell)} \in \mathbb{R}^{L \times C}$ represents the input of the ℓ-th layer for the i-th sample, and L and C are the sequence length and hidden dimension, respectively.

Consider that removing transformer layers contiguously from ℓ^*-th layer to $(\ell^* + n)$-th layer, the $(\ell^* + n)$-th layer takes the input of the ℓ^*-th layer as its own input, i.e.,

$$\mathbf{X}^{(\ell^*+n)} = \mathbf{X}^{(\ell^*)} + f(\mathbf{X}^{(\ell^*)}, \theta^{(\ell^*+n)}). \quad (3)$$

However, we find *layer pruning brings a large gap on the channel magnitude at the pruning interface, which negatively impacts the model performance*, as shown in Figure 1. In Section 3.2 and Section 3.3, we investigate the root causes two aspects, and propose our solutions accordingly.

3.2 Channel Magnitude Alignment

Layer-wise Channel Mismatch. We find that *the mismatch of channel magnitude would directly affect the performance of pruned LLMs*. As shown in Figure 1 (a), the magnitude of hidden states in different layers of LLM varies, while the removal of contiguous layers brings a significant gap between the channel magnitude.

For LLM to remove its layer with index in $[\ell^*, \ell^* + n)$, we statistically compute channel-wise scaling parameters as follows. For activation of each channel k, we compute the ratio of mean activation magnitude between the input of $(\ell^* + n)$-th layer and ℓ^*-th layer over the calibration set and derive the channel-wise scaling parameter $\mathbf{d} \in \mathbb{R}^C$, where d_k is defined as

$$d_k(\ell^*, \ell^* + n) = \frac{1}{BL} \sum_{i=1}^{B} \sum_{j=1}^{L} \frac{|\mathbf{X}_{i,j,k}^{(\ell^*+n)}|}{|\mathbf{X}_{i,j,k}^{(\ell^*)}|}. \quad (4)$$

Quantitative Evaluation. It can be found from Figure 1 (b) that $\mathbf{X}^{(\ell^*)} \cdot \mathbf{d}$ matches the magnitude directly, together with the improved perplexity scores. Additionally, we also perturb the input $\mathbf{X}^{\ell^*}$ around the scaling parameter $\mathbf{d} \in \mathbb{R}^C$ with the perturbation coefficient α, i.e.,

$$\mathbf{X}^{(\ell^*+n)} = \alpha \mathbf{X}^{(\ell^*)} \cdot \mathbf{d} + f(\alpha \mathbf{X}^{(\ell^*)} \cdot \mathbf{d}, \theta^{(\ell^*+n)}).$$

From Figure 1 (b), finding the optimal scaling parameter $\mathbf{d}$ yields lower perplexity than vanilla layer-pruning without scaling. By varying α, the model suffers from the performance drop dramatically.

Remarks. The issue of channel magnitude mismatch has been neglected by the prevalent metric of cosine similarity, given that the scalings over input are cancelled out, i.e., $\cos(\mathbf{a}, \mathbf{b}) = \cos(x\mathbf{a}, y\mathbf{b})$ for any $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and $x, y \in \mathbb{R}^+$.

3.3 Token Magnitude Smoothing

Token-wise Scaling Mismatch. As suggested by recent research (Liu et al., 2024b; Xiao et al., 2023b), there are massive outliers specific to particular tokens (e.g., `[BOS]` and delimiter tokens) with magnitudes over 1e3, as shown in Figure 1 (c).

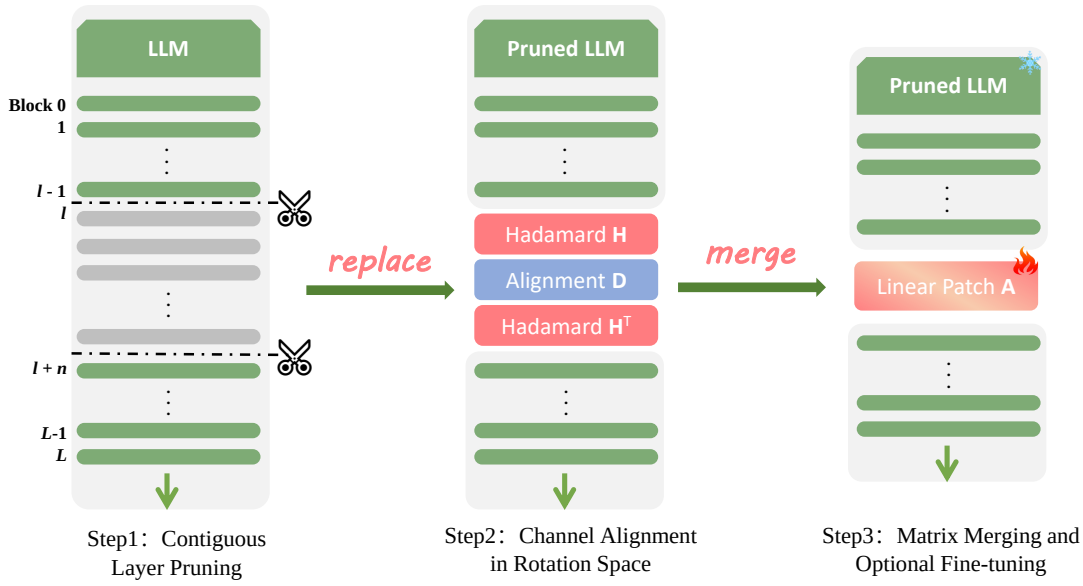


Figure 2: Overview of LINEARPATCH. In step1, we remove layers contiguously with cosine similarity metric. In step2, we obtain channel-wise scaling parameters in the Hadamard rotation space. In step3, we fuse the channel-wise scaling parameters and the Hadamard transformations to obtain LINEARPATCH, which allows further fine-tuning.

Consequently, *the scaling parameter d_k may not fit all tokens along the k-th channel, given the tokens within this channel also exhibit large variance of magnitude.* Specifically, we calculate the standard deviations of magnitude gap averaged over all tokens, i.e.,

$$\sigma_{\mathbf{d}}(\ell^*, \ell^* + n) = \frac{1}{C}\sum_{k=1}^{C} \sigma\left(\frac{|\mathbf{X}_k^{(\ell^*+n)}|}{|\mathbf{X}_k^{(\ell^*)}|}\right), \quad (5)$$

where $\sigma(\cdot)$ is the standard deviation. A small $\sigma_{\mathbf{d}}$ is usually preferred to share the scaling parameter for all tokens. However, due to the presence of outliers, we observe $\sigma_{\mathbf{d}} = 2137.75$ when pruning 9 layers from LLaMA-2-7B.

Hadamard Transformation. Inspired by recent research (Lin et al., 2024; Liu et al., 2024c; Ashkboos et al., 2024b; Sun et al., 2024b), Hadamard transformation over activations can effectively suppress outliers, as shown in Figure 1 (d). A Walsh-Hadamard matrix (Horadam, 2012) is a specific type of Hadamard matrix with size $C = 2^n$, which can be constructed recursively as:

$$\begin{cases} \mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ \mathbf{H}_{2^n} = \mathbf{H}_2 \otimes \mathbf{H}_{2^{n-1}}, \end{cases} \quad (6)$$

whereas for cases $C \neq 2^n$, we follow Ashkboos et al. (2024b) to factorize $C = 2^n m$ and use a Kronecker construction $\mathbf{H}_C = \mathbf{H}_{2^n} \otimes \mathbf{H}_m$. The orthogonality of Hadamard matrix (i.e., $\mathbf{H}^\top\mathbf{H} = \mathbf{I}$) makes the following transformations equivalent:

$$\mathbf{X}^{(\ell^*)} = (\mathbf{X}^{(\ell^*)}\mathbf{H})\mathbf{H}^\top, \quad (7)$$

where $\mathbf{X}^{(\ell^*)}\mathbf{H}$ denotes the rotated activations of ℓ^*-th layer. As shown in Figure 1 (d), the rotation thus effectively redistributes outliers among all channels. With the rotated activations, it is thus more friendly to share the same scaling parameter $\mathbf{d}$ for all tokens, with $\sigma_{\mathbf{d}}$ down to 230.32.

3.4 LINEARPATCH: the Ultimate Recipe

From Figure 2, LINEARPATCH can be obtained by fusing the proposed channel-wise scaling $\mathbf{d}$ and Hadamard transformations $\mathbf{H}$, which *both reduces the transformation overhead and allows efficient fine-tuning to enhance the pruned LLMs*.

Formulation under Spectral Theorem. Combining Section 3.2 and Section 3.3, we first apply the Hadamard transformation on the input $\mathbf{X}^{(\ell^*)}$, over which we apply the channel-wise scaling on the rotated space. A key ingredient of the LINEARPATCH is to fuse all operations into a single matrix $\mathbf{A}$, i.e.,

$$\mathbf{X_{new}}^{(\ell^*)} = \mathbf{X}^{(\ell^*)}\mathbf{H}\mathbf{D}\mathbf{H}^\top = \mathbf{X}^{(\ell^*)}\mathbf{A}, \quad (8)$$

where $\mathbf{D} = \mathrm{diag}(\mathbf{d}) \in \mathbb{R}^{C\times C}$ and $\mathbf{A} = \mathbf{H}\mathbf{D}\mathbf{H}^\top$. The last equality stems from the spectral theorem (Helson, 2006), which states that any real symmetric matrix can be decomposed into an orthogonal

matrix of eigenvectors (i.e., $\mathbf{H}$) and a diagonal matrix of real eigenvalues (i.e., $\mathbf{D}$).

As shown in Figure 2, LINEARPATCH is a plug-and-play method by inserting $\mathbf{A}$ into the pruned layers, which effectively bridges the gap induced by layer-pruning. Additionally, this also reduces the inference overhead by launching only a single GEMM operation for the matrix multiplication instead of three. Overall LINEARPATCH has negligible overhead during the inference and more latency analysis can be found in 6.

Offline Knowledge Distillation. Initialized from Equation (8), we show that LINEARPATCH can be efficiently fine-tuned via offline knowledge distillation to enhance the pruned LLMs. Specifically, given a tiny training corpus $\mathbf{X} \in \mathcal{T}$ (e.g., only 5,000 training samples), we collect the output logits $\mathbf{o}_t \in \mathcal{R}^{B\times L\times V}$ of the original teacher model, where B, L, V denote the batch size, sequence length, and the vocabulary size, respectively. However, saving the entire $\mathbf{o}_t$ is extremely memory consume due to the huge vocabulary size. To reduce the storage, we save top-K logits $\mathbf{o}_t^K \in \mathcal{R}^{B\times L\times K}$ with their indices from the vocabulary, which helps to focus on the most informative candidate tokens. Meanwhile, we also collect the output logits $\mathbf{o}_s^K \in \mathbb{R}^{B\times L\times K}$ from the student model (i.e., the pruned LLM), with the same indices of the vocabulary from the teacher model. Thus we minimize the Kullback-Leibler (KL) divergence (Chen et al., 2024c,b) between the output distributions as

$$\min_{\mathbf{A}} \mathbb{E}_{\mathbf{X}\in\mathcal{T}} \ \mathbf{KL}\left(z(\mathbf{o}_t^K)), z(\mathbf{o}_s^K)\right), \qquad (9)$$

where $z(\cdot)$ indicates the softmax function. By freezing all the rest parameters except for the linear patch $\mathbf{A}$, the training paradigm sufficiently reduces the memory consumption and a single 24GB GPU can meet the fine-tuning of LLMs with size 7B within 30 minutes.

Apart from knowledge distillation with Kullback-Leibler divergence loss, we also tried to minimize the mean squared error (MSE) of the pruned layers following (Chen et al., 2024a). However, the MSE empirically leads to over-fitting and thus brings sub-optimal solutions.

4 Experiments

4.1 Setup

Models. We evaluate the proposed LINEARPATCH on various open-sourced LLMs, including LLaMA-2-7B/13B (Touvron et al., 2023), and LLaMA-3-8B (Dubey et al., 2024). Download links to officially released LLMs can be found in Table 8 in the Appendix B.

Baselines. We mainly compare our methods with state-of-the-art layer-wise pruning approaches with different pruning metrics, including the gradient-base LLM-Pruner (Ma et al., 2023), perplexity-based SLEB (Song et al., 2024), cosine similarity-based ShortGPT (Men et al., 2024) and LLM-Streamline (Chen et al., 2024a). For LLM-Pruner, we use block-wise pruning configuration with C4 calibration set to reproduce all results. For SLEB, we use the official code [1] or the indices of pruned-layers for reproduction. For LLM-Streamline and ShortGPT, we reproduce the methods following the description of the publications.

Evaluation. Three benchmarks are used for evaluation: perplexity (PPL), Massive Multitask Language Understanding (MMLU), and commonsense question answering (QA). For PPL, we measure language generation on WikiText2 (Merity et al., 2016), C4 (Raffel et al., 2020), and PTB (Marcus et al., 1993) datasets. For MMLU, we test five-shot performance on MMLU datasets (Hendrycks et al., 2020). For QA, we assess on 9 commonsense QA tasks and report the average (AVG) for PPL, weighted average (Weighted AVG) for MMLU, and both AVG and retained performance (RP) for QA. More details can be found in Appendix C.

4.2 Implementation Details

Calibration and Fine-tuning. To decide the pruned layer and initialize the channel-wise scaling parameter, we randomly select 128 sentences with sequence length 2048 from WIKITEXT-2 datasets for all models as few-shot calibration samples. We conducted ablation on the size of calibration set in Appendix D. To fine-tune the LINEARPATCH, we use AdamW optimizer with learning rate 1e-4, using 5000 sentences with sequence length 2048 from WIKITEXT-2 datasets for 1 epoch.

Resource Dependence. Our implementation is based on the PyTorch framework. All experiments are conducted on a server with a 24GB GPU. For model size 7B, the initialization process of LINEARPATCH is about 30 seconds and the optional fine-tuning process is about 30 minutes.

[1] https://github.com/jiwonsong-dev/SLEB

Model	n/N	Method	Ratio	ARC-c	ARC-e	BoolQ	HeSw	PIQA	WG	WSC	Race-h	CoPa	AVG	RP
LLaMA-2-7B	0/32	Dense	-	46.25	74.58	77.74	75.97	79.11	68.98	80.59	39.62	87.00	69.98	100
	9/32	LLMPruner	26.99	31.91	52.90	62.42	54.41	71.33	53.20	65.57	28.52	79.00	55.47	78.14
	9/32	SLEB	27.03	31.91	52.31	46.09	58.28	69.59	58.25	69.23	32.25	79.00	55.21	78.41
	9/32	ShortGPT	27.03	32.76	48.61	62.17	56.17	64.36	64.33	71.06	32.25	77.00	56.52	80.29
	9/32	LLM-Streamline(None)	27.03	32.76	48.61	62.17	56.17	64.36	64.33	71.06	32.25	77.00	56.52	80.29
	9/32	LINEARPATCH$_{[S/L]}$	26.78	33.45	55.22	62.14	57.67	67.46	65.11	77.29	34.93	79.00	**59.14**	**84.08**
	7/32	LLMPruner	20.56	35.24	60.61	62.42	61.66	75.41	54.78	71.43	31.67	80.00	59.25	83.80
	7/32	SLEB	21.02	33.02	56.57	63.91	62.49	73.07	58.96	69.23	32.06	84.00	59.26	83.66
	7/32	ShortGPT	21.02	36.18	55.89	62.17	62.66	70.40	65.98	77.29	33.78	81.00	60.59	86.06
	7/32	LLM-Streamline(None)	21.02	36.18	55.89	62.17	62.66	70.40	65.98	77.29	33.78	81.00	60.59	86.06
	7/32	LINEARPATCH$_{[S/L]}$	20.78	37.63	61.24	62.14	63.49	70.46	65.90	79.49	36.46	85.00	**62.42**	**88.88**
LLaMA-2-13B	0/40	Dense	-	49.06	77.40	80.61	79.35	80.52	72.38	86.81	40.48	91.00	73.07	100
	10/40	LLMPruner	23.90	39.51	67.26	65.84	72.24	77.91	57.30	73.26	33.59	85.00	63.55	86.32
	10/40	SLEB	24.37	39.93	66.04	66.76	68.24	75.63	63.61	75.46	36.94	84.00	64.07	87.54
	10/40	ShortGPT	24.37	43.00	63.51	58.20	69.29	72.52	69.85	81.32	36.84	87.00	64.61	88.45
	10/40	LLM-Streamline(None)	24.37	43.00	63.51	58.20	69.29	72.52	69.85	81.32	36.84	87.00	64.61	88.45
	10/40	LINEARPATCH$_{[S/L]}$	24.17	44.20	65.53	62.39	70.15	73.83	69.61	81.68	38.09	89.00	**66.05**	**90.49**
	8/40	LLMPruner	19.48	41.98	67.51	63.33	68.76	76.50	56.51	68.86	32.06	85.00	62.28	84.78
	8/40	SLEB	19.50	36.43	61.83	62.32	67.03	75.08	62.51	78.02	34.83	83.00	62.34	84.74
	8/40	ShortGPT	19.50	44.03	67.38	57.00	72.38	75.24	69.61	79.85	38.18	89.00	65.85	90.27
	8/40	LLM-Streamline(None)	19.50	44.03	67.38	57.00	72.38	75.24	69.61	79.85	38.18	89.00	65.85	90.27
	8/40	LINEARPATCH$_{[S/L]}$	19.30	44.54	69.53	67.74	73.02	75.68	69.38	82.78	39.71	92.00	**68.26**	**93.45**
LLaMA-3-8B	0/32	Dense	-	53.41	77.78	81.28	79.16	80.85	72.85	86.45	40.19	89.00	73.44	100
	7/32	LLMPruner	19.37	35.32	59.30	55.23	51.48	72.58	59.98	67.03	31.39	81.00	57.03	77.12
	7/32	SLEB	19.01	34.04	60.06	45.17	62.01	74.05	55.01	67.40	32.82	45.17	52.86	72.48
	7/32	ShortGPT	19.01	42.41	56.65	65.26	64.70	70.89	71.19	73.63	34.16	75.00	61.54	83.79
	7/32	LLM-Streamline(None)	19.01	28.92	39.56	38.07	33.26	59.47	55.56	59.71	24.02	60.00	44.29	59.99
	7/32	LINEARPATCH$_{[S]}$	18.80	43.17	60.82	75.66	66.74	72.85	70.17	75.82	37.51	77.00	**64.42**	**87.82**
	7/32	LINEARPATCH$_{[L]}$	18.80	34.39	51.26	57.52	49.31	63.33	63.22	72.53	29.95	67.00	54.28	73.57
	5/32	LLMPruner	13.39	39.51	68.10	71.28	64.69	76.33	64.48	74.36	35.60	78.00	63.59	86.23
	5/32	SLEB	13.58	39.68	66.16	54.71	67.39	75.90	62.51	73.63	34.16	83.00	61.90	83.88
	5/32	ShortGPT	13.58	45.56	63.51	73.12	70.13	74.92	71.19	75.09	36.94	79.00	65.50	89.27
	5/32	LLM-Streamline(None)	13.58	47.35	66.20	73.52	71.10	74.27	71.03	76.56	36.65	84.00	66.74	90.84
	5/32	LINEARPATCH$_{[S]}$	13.37	45.73	68.60	73.30	70.71	76.01	73.09	79.85	38.18	82.00	67.50	91.91
	5/32	LINEARPATCH$_{[L]}$	13.37	48.55	70.71	74.25	72.52	76.71	73.95	81.32	38.37	86.00	**69.15**	**94.15**

Table 1: Comparison on QA benchmark with training-free methods. n denotes the number of pruned layers and N denotes the total number of layers of the model. The Ratio column represents the proportion(%) of pruning parameters to the total parameters of the model. AVG column denotes the average accuracy(%) and RP column denotes the retained performance(%). Same interpretation is adopted in all the tables.

Pruning Configurations. We set pruning ratios below 30% to be consistent with prior studies. Detailed pruning configurations, including the number and indices of pruned layers for each method, are provided in Table 9 in the Appendix B.

4.3 Main Results

Our LINEARPATCH can be seamlessly integrated with layer-wise pruning methods that remove contiguous layers. We denote LINEARPATCH combined with ShortGPT and LLM-Streamline pruning methods as LINEARPATCH$_{[S]}$ and LINEARPATCH$_{[L]}$, respectively. In most cases, they prune the same set of layers and are collectively labeled as LINEARPATCH$_{[S/L]}$.

4.3.1 Comparison on Training-free Methods

We first evaluate the performance in a training-free setting on commonsense QA benchmarks. For the sake of fairness, all the methods do not include fine-tuning. For LLM Pruner, we discard the process of fine-tuning using LoRA. For LLM-Streamline, according to its publication, discarding layer replacement and the process of offline distillation is denoted by LLM-Streamline(None). For results of LINEARPATCH on more models and benchmarks, please refer to Appendix E.

As shown in Table 1, LINEARPATCH achieves superior performance on QA benchmark evaluation compared to other methods. Specifically, on the LLaMA-2-7B model with a 7/32 pruning ratio, LINEARPATCH achieves the retained performance ratio of 88.88%, outperforming LLMPruner (83.80%) and SLEB (83.66%). On the LLaMA-3-8B model with a 5/32 pruning ratio, LINEARPATCH achieves the retained performance ratio of 94.15%, significantly higher than LLMPruner (86.23%) and SLEB (83.88%). When ShortGPT/LLM-

Model	n/N	Method	Ratio	ARC-c	ARC-e	BoolQ	HeSw	PIQA	WG	WSC	Race-h	CoPa	AVG	RP
LLaMa2-7B	0/32	Dense	0	46.25	74.58	77.74	75.97	79.11	68.98	80.59	39.62	87.00	69.98	100
	7/32	LLM-Streamline(FFN) + FT	19.01	38.23	60.48	70.18	63.75	69.86	67.48	80.95	37.51	79.00	63.05	90.00
	7/32	LINEARPATCH$_{[L]}$	20.78	37.63	61.24	62.14	63.49	70.46	65.90	79.49	36.46	85.00	62.42	88.88
	7/32	LINEARPATCH$_{[L]}$ + FT	20.78	38.23	64.35	65.32	69.33	73.23	67.40	83.88	38.37	87.00	**65.23**	**92.83**
LLaMA-3-8B	0/32	Dense	0	53.41	77.78	81.28	79.16	80.85	72.85	86.45	40.19	89.00	73.44	100
	5/32	LLM-Streamline(FFN) + FT	11.39	30.03	39.94	65.32	49.19	59.79	67.80	81.32	31.39	71.00	55.09	74.34
	5/32	LINEARPATCH$_{[L]}$	13.37	48.55	70.71	74.25	72.52	76.71	73.95	81.32	38.37	86.00	69.15	94.15
	5/32	LINEARPATCH$_{[L]}$ + FT	13.37	48.12	72.77	70.98	74.63	77.42	74.03	84.62	38.56	89.00	**70.01**	**95.16**

Table 2: Comparison on QA benchmark with SOTA post-training method LLM-Streamline.

Model	n/N	Method	Ratio	WIKI-2	C4	PTB	AVG
LLaMA-2-7B	0/32	Dense	0	5.47	6.97	22.51	11.65
	7/32	LLM-Streamline(FFN) + FT	19.01	9.60	17.10	47.04	24.58
	7/32	LINEARPATCH$_{[L]}$	20.78	13.22	14.58	45.97	24.59
	7/32	LINEARPATCH$_{[L]}$ + FT	20.78	8.09	11.25	32.48	**17.27**
LLaMA-3-8B	0/32	Dense	-	6.14	8.88	10.59	8.54
	5/32	LLM-Streamline(FFN) + FT	11.39	383.15	201.60	101.35	228.70
	5/32	LINEARPATCH$_{[L]}$	13.37	15.13	17.41	19.30	17.28
	5/32	LINEARPATCH$_{[L]}$ + FT	13.37	9.00	13.34	14.34	**12.23**

Table 3: Comparison on PPL benchmark with SOTA post-training method LLM-Streamline.

Streamline is employed as the baselined pruning method, LINEARPATCH$_{[S/L]}$ achieves significant performance improvement compared with its baseline. For example, on the LLaMA-2-7B model with a 9/32 pruning ratio, LINEARPATCH$_{[S/L]}$ achieves the retained performance ratio of 84.08%, leading both ShortGPT and LLM-Streamline with 3.79%. On the LLaMa-13B model with a 8/40 pruning ratio, LINEARPATCH$_{[S/L]}$ achieves the retained performance ratio of 93.45%, leading both ShortGPT and LLM-Streamline with 3.18%.

4.3.2 Comparison on Post-training Methods

We present a comprehensive comparison of our proposed LINEARPATCH method against the state-of-the-art post-training method, LLM-Streamline. For fair comparison, we used the same dataset, samples, and learning rate for fine-tuning.

Results on QA Benchmark. For QA benchmark, results in Table 2 demonstrate the superior performance of LINEARPATCH, especially when combined with fine-tuning. For instance, on LLaMA-2-7B with 7/32 layers pruned, LINEARPATCH$_{[L]}$+FT achieves an average accuracy of 65.23%, outperforming LLM-Streamline (63.05%). Similarly, on LLaMA-3-8B with 5/32 layers pruned, LINEARPATCH$_{[L]}$+FT attains an average accuracy of 70.01%, significantly outperforming LLM-Streamline, which yields suboptimal results due to its randomly initialized replacement networks. Notably, our method achieved a retained performance (RP) of 95.16% on LLaMA-3-8B with 5/32 layers pruned.

Results on PPL Benchmark. For PPL benchmark, LINEARPATCH$_{[L]}$ consistently outperforms or matches LLM-Streamline, as shown in Table 3. For instance, on LLaMA-2-7B with 7/32 layers pruned, LINEARPATCH$_{[L]}$+FT achieves an average perplexity of 17.27, significantly outperforming LLM-Streamline (24.58). Similarly, on LLaMA-3-8B with 5/32 layers pruned, LINEARPATCH$_{[L]}$+FT attains an average perplexity of 12.23, demonstrating its robustness and effectiveness.

4.4 Discussions

Tunable parameters and Loss Functions. We conducted a series of comprehensive experiments on LLaMA-2-7B with 9/32 layers pruned to systematically evaluate how different training configurations influence the performance of LINEARPATCH, as detailed in Table 4. The results indicate that the choice of training settings significantly affects the performance of LINEARPATCH. Specifically, LINEARPATCH with model logits as the distillation target and the KL divergence loss function achieved the best overall performance. This configuration attained the lowest average perplexity (19.58) on language modelling tasks and the highest average accuracy (61.71%) on QA tasks, with a relative performance improvement of 2.13% over LLM-Streamline. Our analysis suggests that using replaced layer output for distillation may lead to overfitting issues and inferior performance. Fur-

Model	Parameters	Distillation Target	Loss	Ratio	WIKI-2	C4	PTB	PPL AVG	QA AVG	QA RP
Dense	-	-	-	-	5.47	6.97	22.51	11.65	69.98	100
LLM-Streamline(FFN) + FT	FFN	Replaced layer ouput	MSE	25.02	13.00	27.22	68.97	36.40	59.44	84.74
LLM-Streamline(FFN) + FT	FFN	Model logits	KL	25.02	-	-	-	-	-	NaN
LINEARPATCH$_{[L]}$ + FT	LINEARPATCH	Replaced layer ouput	MSE	26.78	15.26	19.54	54.33	29.71	59.58	84.80
LINEARPATCH$_{[L]}$ + FT	Diagonal	Model logits	KL	26.78	17.51	18.64	51.64	29.26	59.40	84.45
LINEARPATCH$_{[L]}$ + FT	LINEARPATCH	Model logits	KL	26.78	8.60	12.98	37.16	**19.58**	**61.71**	**88.15**

Table 4: Comparisons on tunable parameters, distillation target and loss functions. FT denotes fine-tuning.

	WIKI-2	C4	PTB	AVG	ARC-c	ARC-e	BoolQ	HeSw	PIQA	WG	WSC	Race-h	CoPa	AVG	RP
Dense	5.47	6.97	22.51	11.65	46.25	74.58	77.74	75.97	79.11	68.98	80.59	39.62	87.00	69.98	100
Vanilla	35.68	36.10	96.52	56.10	32.76	48.61	62.17	56.17	64.36	64.33	71.06	32.25	77.00	56.52	80.29
$+\mathbf{d}$	20.68	22.75	57.67	33.70	35.07	54.97	62.17	56.93	66.76	63.77	75.09	34.83	78.00	58.62	83.56
$+\mathbf{A}$	18.60	19.28	53.00	**30.29**	33.53	55.22	62.14	57.69	67.41	65.04	77.29	34.93	79.00	59.14	84.09
+ FT	8.60	12.98	37.16	**19.58**	34.81	60.65	62.48	64.52	70.29	66.69	76.92	39.04	80.00	**61.71**	**88.15**

Table 5: Ablation study on the ingredients of LINEARPATCH over LLaMA-2-7B with 9/32 layers pruned. $+\mathbf{d}$ applies channel scaling, $+\mathbf{A}$ (i.e., $\mathbf{HDH}^\top$) refers to the LINEARPATCH, and +FT denotes fine-tuning with knowlede distillation. Note that we omit ablating $\mathbf{H}$ since Hadamard transformation alone is an equivalent operation.

thermore, LLM-Streamline's approach of incorporating a randomly initialized lightweight network introduces instability during logits distillation, potentially causing performance degradation and diverse training. These findings highlight that LINEARPATCH offers a more effective initialization to enable more reliable performance recovery through lightweight training paradigms.

The Ingredients of Linear Patch. Table 5 ablates the impact of components of the proposed LINEARPATCH. The results show that each component of LINEARPATCH contributes significantly to the overall performance. Pruned model suffer from substantial performance degradation, with an average perplexity of 56.10 and a retained performance of 80.29% on QA tasks. When scaling parameters $\mathbf{d}$ is introduced, the performance improves, with an average perplexity of 33.70 and a retained performance of 83.56%. Further incorporating Hadamard rotation to create LINEARPATCH $\mathbf{A}$ achieves an average perplexity of 30.29 and a retained performance of 84.09%, with a relative performance improvement of 3.8% over the baseline. Fine-tuning can yield significant additional benefits, with a relative retained performance gain of 4.06%.

Inference Overhead. As LINEARPATCH is a single linear layer inserted at the pruning interface, there is empirically no difference on the end-to-end inference latency. Here we study the overhead of LINEARPATCH alone and compare it with the existing baseline, i.e., LLM-Streamline with feed-forward-network. We conduct experiments using a batch size of 16 and a sequence length of 2048, with 1000 iterations to measure the average inference time for the single additional layer. The hidden size was set to 4096, aligning with the configurations of LLaMA-2-7B. All calculations are based on float16 precision. As shown in Table 6, LINEARPATCH is **~8x** smaller in size and **~190x** faster than LLM-Streanline (FFN). We also conducted overhead analysis on offline storage overhead in Appendix A.

Method	Parameters (M)	Time cost (s)
LLM-Streamline (FFN)	135.27	0.094927
LINEARPATCH	16.78	0.000576

Table 6: Online inference time overhead.

5 Conclusion

In conclusion, we introduce LINEARPATCH, a simple yet effective plug-and-play technique that addresses the critical issue of activation magnitude mismatch in layer-pruned large language models (LLMs). By leveraging the Hadamard transformation and channel-wise scaling, LINEARPATCH efficiently aligns activations across layers, significantly enhancing model performance with negligible inference overhead. Extensive empirical evaluations are conducted to demonstrate the effectiveness of the proposed approach. We hope the proposed LINEARPATCH can shed more light on simple and light-weight algorithms of LLM compression without compromising the performance.

6 Limitations

Layer pruning can streamline LLMs for efficiency, but it may unevenly degrade model performance across different tasks. For example, while some text generation tasks might remain robust, complex reasoning or context-dependent tasks could suffer. Although we conduct extensive experiments on several benchmarks, such as PPL, MMLU, and commonsense QA, it might not be enough to assess all abilities of LLMs. Therefore, in future work, a comprehensive evaluation framework is necessary to thoroughly assess the trade-offs between efficiency gains from layer pruning and potential performance degradation across diverse tasks and contexts.

7 Ethics Statement

Layer pruning methods can significantly reduce the computational costs of deploying Large Language Models (LLMs), making them more accessible to a broader range of users. However, these methods do not address the social biases embedded in LLMs, which often stem from the training data and can affect fairness and inclusivity. It is crucial to ensure ethical deployment of LLMs.

References

2019. Winogrande: An adversarial winograd schema challenge at scale.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024a. Slicegpt: Compress large language models by deleting rows and columns. arXiv preprint arXiv:2401.15024.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024b. Quarot: Outlier-free 4-bit inference in rotated llms. arXiv preprint arXiv:2404.00456.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 7432–7439.

Xiaodong Chen, Yuxuan Hu, and Jing Zhang. 2024a. Compressing large language models by streamlining the unimportant layer. arXiv preprint arXiv:2403.19135.

Xinrui Chen, Yizhi Wang, Yao Li, Xitong Ling, Mengkui Li, Ruikang Liu, Minxi Ouyang, Kang Zhao, Tian Guan, and Yonghong He. 2024b. Low bit-width zero-shot quantization with soft feature-infused hints for iot systems. IEEE Internet of Things Journal.

Xinrui Chen, Yizhi Wang, Renao Yan, Yiqing Liu, Tian Guan, and Yonghong He. 2024c. Texq: zero-shot network quantization with texture feature distribution calibration. Advances in Neural Information Processing Systems, 36.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. arXiv preprint arXiv:1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

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

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. arXiv preprint arXiv:2403.17887.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Henry Helson. 2006. The spectral theorem. The Spectral Theorem, pages 23–41.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300.

Kathy J Horadam. 2012. Hadamard matrices and their applications. Princeton university press.

Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. 2024. Sp3: Enhancing structured pruning via pca projection. In Findings of the Association for Computational Linguistics ACL 2024, pages 3150–3170.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. arXiv preprint arXiv:2310.06825.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. arXiv preprint arXiv:2402.02834, 11.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In Thirteenth international conference on the principles of knowledge representation and reasoning.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024. Rotation and permutation for advanced outlier management and efficient quantization of llms. arXiv preprint arXiv:2406.01721.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. 2024b. Intactkv: Improving large language model quantization by keeping pivot tokens intact. arXiv preprint arXiv:2403.01241.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024c. Spinquant–llm quantization with learned rotations. arXiv preprint arXiv:2405.16406.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems, 36:21702–21720.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19(2):313–330.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140):1–67.

Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. 2024. Llama-nas: Efficient neural architecture search for large language models. arXiv preprint arXiv:2405.18377.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4938–4947.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. arXiv preprint arXiv:2402.09025.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. 2024a. Massive activations in large language models. arXiv preprint arXiv:2402.17762.

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

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. 2024b. Flatquant: Flatness matters for llm quantization. arXiv preprint arXiv:2410.09426.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. arXiv preprint arXiv:2501.12599.

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

Tycho FA van der Ouderaa, Markus Nagel, Mart Van Baalen, Yuki M Asano, and Tijmen Blankevoort. 2023. The llm surgeon. arXiv preprint arXiv:2312.17244.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. arXiv preprint arXiv:2310.06694.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning, pages 38087–38099. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023b. Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. arXiv preprint arXiv:2309.10305.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. arXiv preprint arXiv:2407.10671.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. arXiv preprint arXiv:2402.11187.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? arXiv preprint arXiv:1905.07830.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. Plug-and-play: An efficient post-training pruning method for large language models. In The Twelfth International Conference on Learning Representations.

A Offline Storage Overhead Analysis.

As demonstrated in Table 7, assuming that the training samples with batch size B and sequence length L, LINEARPATCH achieves a substantial reduction in offline storage overhead, reaching up to 40× for a hidden size of 4096, compared to LLM-Streamline during offline fine-tuning.

Method	Stored input size	Stored target size
LLM-Streamline	$B \times L \times 4096$	$B \times L \times 4096$
LINEARPATCH	$B \times L$	$B \times L \times 100$

Table 7: Offline storage overhead.

B Details on Pruned Models

We obtain the officially released LLMs from the sources listed in Table 8 for our experiments. Additionally, Table 9 provides a detailed illustration of the specific pruned layer indices for various configurations across different models.

Model	Download Link
LLaMA-2-7B	https://huggingface.co/meta-llama/Llama-2-7B
LLaMA-2-13B	https://huggingface.co/meta-llama/Llama-2-13B
LLaMA-3-8B	https://huggingface.co/meta-llama/Llama-3.1-8B
Baichuan-2-7B	https://huggingface.co/baichuan-inc/Baichuan2-7B-Base

Table 8: Download links to officially released LLMs.

Model	n/N	Method	Ratio	Pruned layer
LLaMA-2-7B	9/32	LINEARPATCH$_{[S/L]}$	26.78%	[21,30)
	7/32	LINEARPATCH$_{[S/L]}$	20.78%	[23,30)
LLaMA-2-13B	10/40	LINEARPATCH$_{[S/L]}$	24.17%	[26,36)
	8/40	LINEARPATCH$_{[S/L]}$	19.30%	[28,36)
LLaMA-3-8B	7/32	LINEARPATCH$_{[S]}$	18.80%	[22,29)
	7/32	LINEARPATCH$_{[L]}$	18.80%	[23,30)
	5/32	LINEARPATCH$_{[S]}$	13.37%	[24,29)
	5/32	LINEARPATCH$_{[L]}$	13.37%	[23,28)
Baichuan-2-7B	9/32	LINEARPATCH$_{[S/L]}$	24.04%	[22,31)
	7/32	LINEARPATCH$_{[S/L]}$	18.65%	[23,30)

Table 9: Details of pruning settings.

C Details of Evaluation Benchmarks

We use a variety of benchmarks for model evaluation, including the perplexity (PPL) benchmark (measured by the average perplexity score), the Massive Multitask Language Understanding (MMLU) benchmark and the question answering (QA) benchmark for model evaluation.

PPL. For PPL benchmarks, we report the perplexity of language generation on WikiText2 (Merity et al., 2016), C4 (Raffel et al., 2020), and PTB (Marcus et al., 1993) datasets.

MMLU. For MMLU benchmark, we test the five-shot performance on the Massively Multitask Language Understanding (MMLU) datasets (Hendrycks et al., 2020).

Commonsense QA. For QA benchmark, we evaluate methods on 9 commonsense QA tasks: 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) (ai2, 2019), WSC273 (WSC) (Levesque et al., 2012), Race-high (Race-h) (Lai et al., 2017) and CoPA (Sarlin et al., 2020). We report the average (AVG) for the PPL benchmark, weighted average (Weighted AVG) for the MMLU benchmark, and average (AVG) as well as the retained performance (RP) for the QA benchmark.

For MMLU benchmark, we use the official code. For PPL and QA benchmarks, we use the lm_eval library from https://github.com/EleutherAI/lm-evaluation-harness.

D Ablation on Size of Calibration Set

We vary the size of the calibration set to evaluate its impact on the performance of LINEARPATCH in Table 10. The results show that a larger calibration set leads to better scaling parameters and improved performance, but the gains diminish beyond a certain size. A calibration set of 128 samples provides a good balance between computational efficiency and performance.

Num of samples	WIKI-2	C4	PTB
64	18.61	19.29	53.03
128	18.60	19.28	53.00
256	18.60	19.28	53.00
512	18.61	19.28	53.01

Table 10: Ablation on the number of calibration samples for scaling parameters statistics.

E Results on More Models and Benchmarks

E.1 Comparison on PPL Benchmarks with Training-free Methods

We also evaluate LINEARPATCH on the Perplexity (PPL) benchmarks, a critical metric for assessing

the language modeling capabilities of pruned models. Lower PPL values signify superior language modeling performance.

As illustrated in Table 11, SLEB with the PPL-based metric demonstrates a slight advantage over other methods whereas it underperforms in the QA benchmarks. Among approaches utilizing cosine similarity-based metrics, the proposed LINEARPATCH consistently surpasses other methods across various models and pruning ratios. For instance, on the LLaMA-2-13B model with an 8/40 pruning ratio, LINEARPATCH achieves an average PPL of 18.10, significantly outperforming LLM-Pruner (35.06) and SLEB (36.61).

Notably, on the LLaMA-3-8B model with a 7/32 pruning ratio, LLM-Streamline nearly fails, yielding an average PPL of 2839.3. In contrast, LINEARPATCH$_{[L]}$ successfully revives the model without additional training, restoring its performance to a functional level. This highlights the robustness of LINEARPATCH and its ability to deliver stable performance improvements across diverse pruning strategies.

E.2 Comparison on MMLU Benchmarks with Training-free Methods

We evaluate the proposed LINEARPATCH method on the MMLU tasks across multiple models in Table 12. Overall, LINEARPATCH demonstrates significant improvements in weighted average accuracy across different models and pruning ratios. For example, it attains weighted average accuracies of 63.84% for LLaMA-3-8B with 5/32 layers pruned, outperforming the best results from other methods. Similarly, on LLaMA-2-13B, it reaches 53.96% and 54.01% for 10/40 and 8/40 layers pruned, respectively, where SLEB almost collapsed in the same case. These results highlight the robustness and effectiveness of LINEARPATCH in enhancing the performance of layer-pruned large language models on MMLU tasks, demonstrating its potential as a simple yet powerful solution for reviving pruned models.

E.3 Results on Baichuan-2-7B in Training-free Case

Besides LLaMA series models, we also provide results on Baichuan-2-7B (Yang et al., 2023) in training-free case to verify the robustness of LINEARPATCH across different model architectures.

Results on PPL Benchmark. Table 13 shows the results on PPL benchmarks on Baichuan-2-7B model. Except for the advance performance of SLEB which uses a PPL-based metric, LINEARPATCH achieves the best performance across the cosine similarity-based metrics methods. Under the pruning ratio of 9/32 and 7/32, LINEARPATCH achieves an average perplexity of 44.50 and 23.80 respectively, significantly lower than the 78.22 and 115.38 achieved by ShortGPT and LLM-Streamline.

Results on QA Benchmark. Table 14 shows the results on QA benchmarks. On the Baichuan-2-7B model with a 7/32 pruning ratio, LINEARPATCH$_{[S/L]}$ achieves a retained performance ratio of 87.27%, leading both ShortGPT and LLM-Streamline with 3.39%. When it comes to the pruning ratio of 9/32, LINEARPATCH$_{[S/L]}$ still maintains the 81.66% of the original performance, outperforming ShortGPT and LLM-Streamline by 4.56%.

Results on MMLU Benchmark. Table 15 shows the results on MMLU benchmarks. LINEARPATCH attains weighted average accuracies of 50.77% and 52.00% for Baichuan-2-7B with 9/32 and 7/32 layers pruned, respectively, significantly outperforming the best results from other methods.

F Visualization of The Magnitude of LLM Layer outputs

See Figure 3 for more visualization of the magnitude of LLM layer output activations. All layer-pruned model exhibit magnitude mismatch.

Model	n/N	Method	Ratio	Metric	WIKI-2	C4	PTB	PPL AVG
LLaMA-2-7B	0/32	Dense	-	-	5.47	6.97	22.51	11.65
	9/32	LLMPruner	26.99	Grad	20.50	16.61	83.02	40.04
	9/32	SLEB	27.03	PPL	11.99	13.93	45.24	**23.72**
	9/32	ShortGPT	27.03	Cos	35.68	36.10	96.52	56.10
	9/32	LLM-Streamline(None)	27.03	Cos	35.68	36.10	96.52	56.10
	9/32	LINEARPATCH$_{[S/L]}$	26.78	Cos	18.60	19.28	53.00	30.29
	7/32	LLMPruner	20.56	Grad	20.50	16.61	83.02	40.04
	7/32	SLEB	21.02	PPL	9.14	11.21	38.45	**19.60**
	7/32	ShortGPT	21.02	Cos	18.45	20.99	62.18	33.87
	7/32	LLM-Streamline(None)	21.02	Cos	18.45	20.99	62.18	33.87
	7/32	LINEARPATCH$_{[S/L]}$	20.78	Cos	13.22	14.58	45.97	24.59
LLaMA-2-13B	0/40	Dense	-	-	4.88	6.47	28.87	13.41
	10/40	LLMPruner	23.90	Grad	9.28	9.87	62.84	27.33
	10/40	SLEB	24.37	PPL	7.60	9.62	69.97	29.06
	10/40	ShortGPT	24.37	Cos	9.77	12.06	49.94	23.92
	10/40	LLM-Streamline(None)	24.37	Cos	9.77	12.06	49.94	23.92
	10/40	LINEARPATCH$_{[S/L]}$	24.17	Cos	8.69	10.70	39.12	**19.50**
	8/40	LLMPruner	19.48	Grad	11.05	11.20	82.93	35.06
	8/40	SLEB	19.50	PPL	8.17	10.07	91.58	36.61
	8/40	ShortGPT	19.50	Cos	8.30	10.36	44.96	21.21
	8/40	LLM-Streamline(None)	19.50	Cos	8.30	10.36	44.96	21.21
	8/40	LINEARPATCH$_{[S/L]}$	19.30	Cos	7.63	9.58	37.08	**18.10**
LLaMa-3-8B	0/32	Dense	-	-	6.14	8.88	10.59	8.54
	7/32	LLMPruner	19.37	Grad	15.08	18.54	24.15	19.26
	7/32	SLEB	19.01	PPL	13.12	16.76	21.04	**16.97**
	7/32	ShortGPT	19.01	Cos	57.76	50.13	67.39	58.43
	7/32	LLM-Streamline(None)	19.01	Cos	2287.73	1491.37	4738.81	2839.30
	7/32	LINEARPATCH$_{[S]}$	18.80	Cos	25.67	28.38	31.22	28.42
	7/32	LINEARPATCH$_{[L]}$	18.80	Cos	69.82	96.68	88.79	85.10
	5/32	LLMPruner	13.39	Grad	10.33	13.79	15.68	13.27
	5/32	SLEB	13.58	PPL	9.88	13.47	16.37	**13.24**
	5/32	ShortGPT	13.58	Cos	27.33	27.06	31.81	28.73
	5/32	LLM-Streamline(None)	13.58	Cos	21.14	24.13	37.41	27.56
	5/32	LINEARPATCH$_{[S]}$	13.37	Cos	16.51	19.42	20.18	18.70
	5/32	LINEARPATCH$_{[L]}$	13.37	Cos	15.13	17.41	19.30	17.28

Table 11: Comparison on PPL benchmark with training-free methods.

Model	n/N	Method	Ratio	STEM	Humanities	Social Sciences	Others	Weighed AVG
LLaMA-2-7B	0/32	Dense	-	36.98	43.25	51.77	52.47	45.90
	9/32	SLEB	27.03	26.31	25.18	27.75	28.19	26.68
	9/32	ShortGPT	27.03	36.88	41.00	50.73	50.96	44.54
	9/32	LLM - Streamline(None)	27.03	36.88	41.00	50.73	50.96	**44.54**
	9/32	LINEARPATCH$_{[S/L]}$	26.78	34.76	40.38	49.89	50.00	43.48
	7/32	SLEB	21.02	26.47	25.10	25.18	29.02	26.32
	7/32	ShortGPT	21.02	31.75	37.90	44.72	46.18	39.98
	7/32	LLM - Streamline(None)	21.02	31.75	37.90	44.72	46.18	39.98
	7/32	LINEARPATCH$_{[S/L]}$	20.78	31.71	39.26	45.82	47.07	**40.88**
LLaMA-2-13B	0/40	Dense	-	44.14	54.35	63.44	60.80	55.63
	10/40	SLEB	24.37	29.49	32.48	34.02	35.13	32.78
	10/40	ShortGPT	24.37	43.20	50.41	62.98	60.95	54.05
	10/40	LLM - Streamline(None)	24.37	43.20	50.41	62.98	60.95	**54.05**
	10/40	LINEARPATCH$_{[S/L]}$	24.17	42.91	50.31	62.43	61.54	53.96
	8/40	SLEB	19.50	27.40	27.56	28.44	30.57	28.41
	8/40	ShortGPT	19.50	42.80	50.13	62.78	61.19	53.88
	8/40	LLM - Streamline(None)	19.50	42.80	50.13	62.78	61.19	53.88
	8/40	LINEARPATCH$_{[S/L]}$	19.30	43.07	50.61	62.56	61.04	**54.01**
LLaMA-3-8B	0/32	Dense	-	55.20	59.00	75.95	71.56	64.80
	7/32	SLEB	19.01	28.69	23.72	29.57	28.62	27.20
	7/32	ShortGPT	19.01	50.27	57.56	73.19	68.57	**61.96**
	7/32	LLM - Streamline(None)	19.01	32.24	38.70	47.03	40.06	39.45
	7/32	LINEARPATCH$_{[S]}$	18.80	45.96	51.90	66.98	63.11	56.52
	7/32	LINEARPATCH$_{[L]}$	18.80	37.61	40.57	46.31	49.23	43.19
	5/32	SLEB	13.58	30.25	25.50	31.82	30.60	29.08
	5/32	ShortGPT	13.58	46.92	53.92	65.65	65.42	57.64
	5/32	LLM - Streamline(None)	13.58	53.47	56.08	74.58	68.32	62.40
	5/32	LINEARPATCH$_{[S]}$	13.37	44.67	50.31	65.91	61.91	55.19
	5/32	LINEARPATCH$_{[L]}$	13.37	54.24	57.15	75.40	71.50	**63.84**

Table 12: Comparison on MMLU benchmark with training-free methods.

Model	n/N	Method	Ratio	Metric	WIKI-2	C4	PTB	PPL AVG
Baichuan-2-7B	0/32	Dense	-	-	6.03	8.96	18.98	11.32
	9/32	LLMPruner	23.29	Grad	27.83	23.67	134.00	61.83
	9/32	SLEB	24.26	PPL	15.02	21.08	59.56	**31.89**
	9/32	ShortGPT	24.26	Cos	49.88	56.64	128.14	78.22
	9/32	LLM-Streamline(None)	24.26	Cos	49.88	56.64	128.14	78.22
	9/32	LINEARPATCH$_{[S/L]}$	24.04	Cos	24.36	32.42	76.71	44.50
	7/32	LLMPruner	18.47	Grad	17.54	18.75	81.54	39.28
	7/32	SLEB	18.87	PPL	10.78	16.09	42.08	**22.98**
	7/32	ShortGPT	18.87	Cos	103.5	152.05	90.58	115.38
	7/32	LLM-Streamline(None)	18.87	Cos	103.5	152.05	90.58	115.38
	7/32	LINEARPATCH$_{[S/L]}$	18.65	Cos	13.87	19.08	38.44	23.80

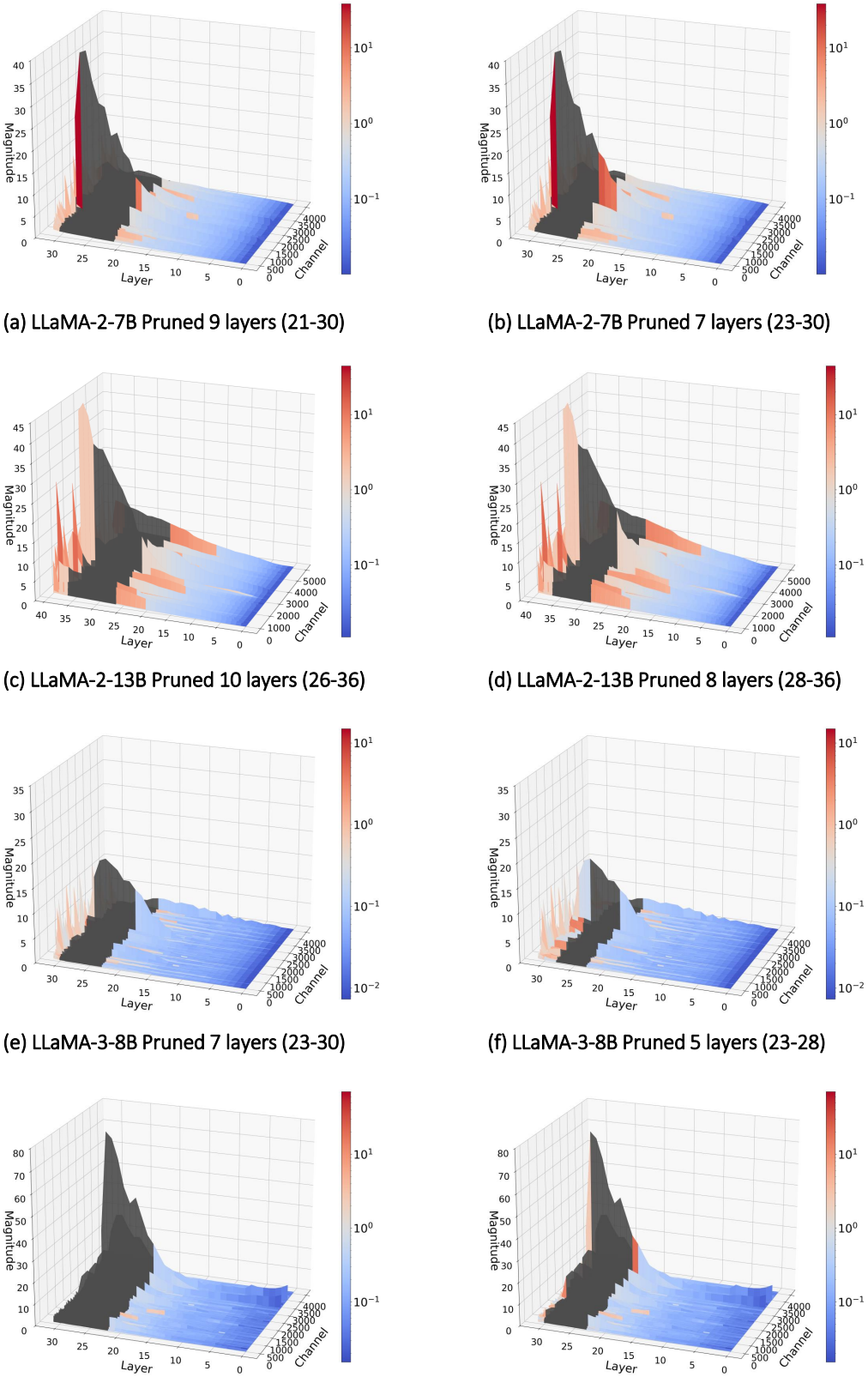
Table 13: Comparison on PPL benchmark with training-free methods on Baichuan-2-7B.

Model	n/N	Method	Ratio	ARC-c	ARC-e	BoolQ	HeSw	PIQA	WG	WSC	Race-h	CoPa	AVG	RP
Baichuan-2-7B	0/32	Dense	-	42.49	72.98	73.91	72.19	77.20	68.43	79.85	38.28	85.00	67.81	100
	9/32	LLMPruner	23.29	32.42	56.36	59.82	54.11	69.70	53.20	59.34	28.04	78.00	54.55	79.64
	9/32	SLEB	24.26	29.18	48.91	62.29	52.14	68.88	55.09	66.30	30.43	75.00	54.25	79.19
	9/32	ShortGPT	24.26	28.67	42.55	67.19	47.09	62.68	62.19	69.23	29.38	65.00	52.66	77.10
	9/32	LLM-Streamline(None)	24.26	28.67	42.55	67.19	47.09	62.68	62.19	69.23	29.38	65.00	52.66	77.10
	9/32	LINEARPATCH$_{[S/L]}$	24.04	30.80	50.04	62.45	52.31	65.72	65.11	71.43	31.58	72.00	**55.72**	**81.66**
	7/32	LLMPruner	18.47	36.86	62.63	62.23	61.25	72.03	54.06	63.74	29.00	80.00	57.98	84.85
	7/32	SLEB	18.87	31.31	55.39	65.47	56.93	71.65	59.12	72.89	33.21	73.00	57.66	84.46
	7/32	ShortGPT	18.87	34.90	51.81	62.39	55.27	65.56	64.72	74.73	31.77	72.00	57.02	83.88
	7/32	LLM-Streamline(None)	18.87	34.90	51.81	62.39	55.27	65.56	64.72	74.73	31.77	72.00	57.02	83.88
	7/32	LINEARPATCH$_{[S/L]}$	18.65	35.15	57.20	62.91	59.02	68.55	66.61	76.19	34.45	73.00	**59.23**	**87.27**

Table 14: Comparison on QA benchmark with training-free methods on Baichuan-2-7B.

Model	n/N	Method	Ratio	STEM	Humanities	Social Sciences	Others	Weighed AVG
Baichuan-2-7B	0/32	Dense	-	44.53	51.30	61.23	60.85	54.23
	9/32	SLEB	24.26	29.03	26.37	28.37	28.50	27.87
	9/32	ShortGPT	24.26	39.07	40.66	50.70	50.31	44.74
	9/32	LLM-Streamline(None)	24.26	39.07	40.66	50.70	50.31	44.74
	9/32	LINEARPATCH$_{[S/L]}$	24.04	42.25	46.80	58.73	56.90	**50.77**
	7/32	SLEB	18.87	33.00	30.67	37.70	37.23	34.23
	7/32	ShortGPT	18.87	42.01	45.48	58.17	55.71	49.88
	7/32	LLM-Streamline(None)	18.87	42.01	45.48	58.17	55.71	49.88
	7/32	LINEARPATCH$_{[S/L]}$	18.65	42.84	48.42	59.60	58.70	**52.00**

Table 15: Comparison on MMLU benchmark with training-free methods on Baichuan-2-7B.



(a) LLaMA-2-7B Pruned 9 layers (21-30)

(b) LLaMA-2-7B Pruned 7 layers (23-30)

(c) LLaMA-2-13B Pruned 10 layers (26-36)

(d) LLaMA-2-13B Pruned 8 layers (28-36)

(e) LLaMA-3-8B Pruned 7 layers (23-30)

(f) LLaMA-3-8B Pruned 5 layers (23-28)

(g) Baichuan-2-7B Pruned 9 layers (22-31)

(h) Baichuan-2-7B Pruned 7 layers (23-30)

Figure 3: Visualization of the magnitude of LLM layer output activations, where pruned layers are represented in grey. All layer-pruned model exhibit magnitude mismatch.