Ghosted Layers: Unconstrained Activation Alignment for Recovering Layer-Pruned LLMs

Anonymous Authors[1]

Abstract

Layer pruning removes entire Transformer decoder blocks from large language models, but introduces a mismatch between the hidden state received by the next surviving layer and the distribution it was trained to process, leading to significant performance degradation. We propose Ghosted Layers, a training-free recovery module that addresses this issue by solving a boundary activation alignment problem. Our method derives a closed-form optimal linear operator from a small calibration set to reconstruct the activation discrepancy introduced by the pruned layers. We show that this solution corresponds to the unconstrained optimum of the alignment objective, whereas existing methods are restricted to constrained solutions over limited operator subspaces. Experiments across multiple LLM backbones and pruning strategies demonstrate that our method consistently improves accuracy and perplexity over prior training-free baselines, while preserving the efficiency gains of layer pruning.

1. Introduction

Large language models (LLMs) have shown strong capabilities across a wide range of natural language tasks (Brown et al., 2020; Touvron et al., 2023a; OpenAI, 2023), but their size makes deployment costly. Various compression techniques have been explored to address this, including unstructured pruning (Frantar & Alistarh, 2023; Sun et al., 2024) and layer pruning (Men et al., 2025; Gromov et al., 2025; Kim et al., 2024; Song et al., 2024). Among these, layer pruning stands out for its practicality, as it removes entire Transformer decoder blocks and yields a smaller model that runs on standard inference stacks without custom kernels or architectural changes. Yet pruned models often exhibit substantial accuracy degradation: pruning a block of layers eliminates the intermediate transformations between the surviving layers, causing a distribution shift at the pruning boundary that propagates through downstream layers. This motivates the need for a recovery mechanism to compensate for the missing layers.

Recent work mitigates this through a variety of training-free recovery modules. ReplaceMe (Shopkhoev et al., 2026) approximates the pruned block's computation with a linear transformation absorbed into the surviving weights, but does not directly address the mismatch at the pruning boundary. In contrast, LinearPatch (Chen et al., 2026) directly targets the boundary activation mismatch, the discrepancy between the hidden state expected by the next surviving layer and the one it actually receives, by inserting a linear operator at the pruning boundary. This operator is implemented as a Hadamard-rotated channel-wise scaling and is symmetric by construction, restricting it to a strict subspace of linear operators and preventing it from reaching the optimum of the alignment objective in the full operator space.

To empirically verify this limitation, we quantify the boundary activation mismatch via the boundary activation error, defined as the mean absolute error (MAE) between the post-boundary hidden state of the original model and the input received by the next layer in the pruned model. Figure 1 compares representative post-pruning recovery methods on the resulting boundary activation error. Existing methods leave a substantial portion of the error uncorrected, suggesting that the boundary activation mismatch is not entirely resolved after recovery.

We address this limitation by deriving the closed-form unconstrained optimum of the boundary activation alignment objective over the full space of linear operators. In contrast, LinearPatch corresponds to a constrained solution restricted to a specific operator subspace. We further show that the optimal operator contains a substantial anti-symmetric component, which is structurally inaccessible to symmetric constructions such as LinearPatch.

Building on this analysis, we propose Ghosted Layers, a training-free recovery module that inserts the closed-form optimal operator at the pruning boundary via a forward hook.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

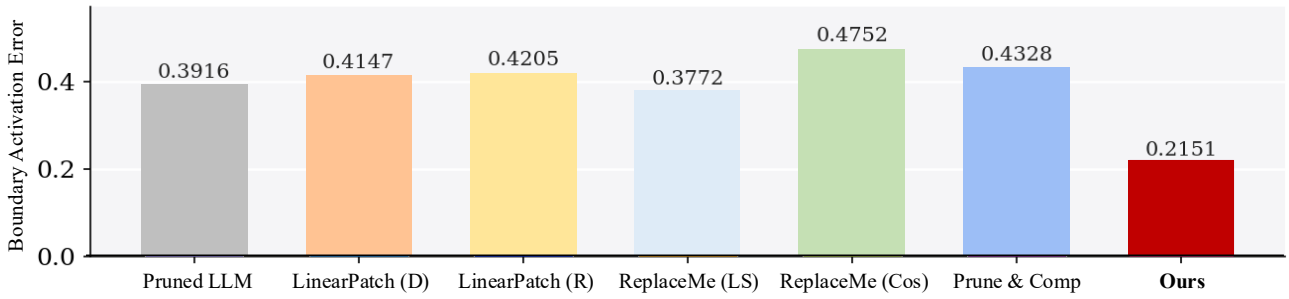

Figure 1. Mean absolute error between the expected boundary activation and the activation received by downstream layers after pruning on LLaMA-3.1-8B. The pruned model is obtained using LLM-Streamline (Chen et al., 2025a) with $n = 7$ layers removed. We compare existing post-pruning recovery methods, including Prune&Comp (Chen et al., 2025b), ReplaceMe (Shopkhoev et al., 2026), and LinearPatch (Chen et al., 2026), against our Ghosted Layers.

It is compatible with any pruning criterion and model architecture, and consistently improves recovery across multiple LLM backbones and benchmarks. These results suggest that solving boundary activation alignment at its unconstrained optimum is sufficient to substantially recover layer-pruned LLMs without retraining.

Contribution of this study can be summarized as follows:

- We formulate post-pruning recovery as an activation alignment problem and derive its closed-form optimal solution from boundary activations.
- We show that this formulation yields an unconstrained optimum that includes a substantial anti-symmetric component, which is inaccessible to symmetric constructions such as LinearPatch.
- We propose Ghosted Layers, a training-free and plug-and-play recovery module compatible with any pruning criterion and model architecture, which consistently outperforms prior methods at matched inference cost.

2. Related works

Structured and layer pruning of LLMs. Structured pruning reduces model size while preserving dense computation, enabling deployment without specialized sparse kernels. Among structured pruning methods, depth pruning removes entire Transformer blocks while preserving the original architecture (Men et al., 2025; Song et al., 2024; Kim et al., 2024; Chen et al., 2025a). Prior work mainly focuses on identifying redundant layers through activation similarity, perplexity, or calibration-based criteria.

Post-pruning recovery. Recent methods attempt to recover the performance degradation caused by layer pruning. Prune&Comp (Chen et al., 2025b) applies channel-wise magnitude compensation, ReplaceMe (Shopkhoev et al., 2026) approximates pruned blocks through linearized block computation, and LinearPatch (Chen et al., 2026) directly repairs the boundary activation mismatch using a constrained symmetric operator. In contrast, our `Ghosted Layers` solves the boundary activation alignment problem over the unconstrained space of linear operators via a closed-form solution. Extended discussions on structured pruning, layer pruning, and post-pruning recovery methods are provided in Appendix A.

3. Method: Ghosted Layers

Layer pruning removes transformer blocks to reduce model size, but the hidden state passed to the next surviving layer no longer matches the one it was trained to process, causing an activation mismatch at the pruning boundary that degrades downstream performance. We propose *Ghosted Layers*, a training-free recovery method that mitigates this mismatch via a closed-form linear operator inserted at each pruning boundary (Figure 2).

3.1. Notation and setup

Let $\mathcal{M} = \{f^{(\ell)}\}_{\ell=0}^{L-1}$ be a pre-trained autoregressive LLM with L Transformer decoder layers, where $f^{(\ell)} : \mathbb{R}^{B\times T\times C} \to \mathbb{R}^{B\times T\times C}$ and B, T, C denote batch size, sequence length, and hidden dimension. Under the standard pre-norm residual architecture:

$$\mathbf{X}^{(\ell+1)} = \mathbf{X}^{(\ell)} + f^{(\ell)}\big(\mathbf{X}^{(\ell)};\, \theta^{(\ell)}\big), \qquad \ell = 0, \dots, L-1. \tag{1}$$

Layer pruning removes a contiguous block $\mathcal{B} = \{\ell^*, \dots, \ell^*+n-1\}$ of n layers, where ℓ^* and ℓ^*+n are the *pre-boundary* and *post-boundary* indices. Unrolling Eq. (1) over $\mathcal{B}$, the post-boundary hidden state in the original model satisfies:

$$\mathbf{X}^{(\ell^*+n)} = \mathbf{X}^{(\ell^*)} + \underbrace{\sum_{k=\ell^*}^{\ell^*+n-1} f^{(k)}\big(\mathbf{X}^{(k)};\, \theta^{(k)}\big)}_{\boldsymbol{\Delta}^{(\ell^*,\, n)}}. \tag{2}$$

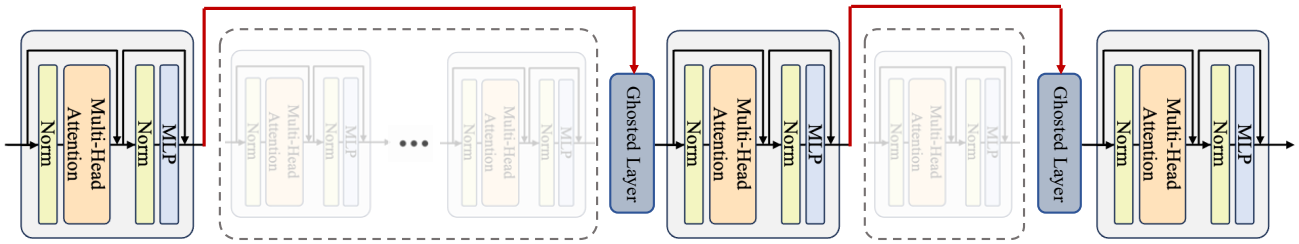

Figure 2. Ghosted Layers as drop-in replacements for pruned transformer blocks. One or more consecutive transformer blocks are removed (dashed), and a single Ghosted Layer (blue) takes their place at the boundary. The red arrows show the hidden state bypassing the pruned region and flowing through the Ghosted Layer before reaching the next surviving block. This applies to both contiguous and non-contiguous pruning, with exactly one Ghosted Layer substituting each pruned region.

The pruned model sets $\mathbf{X}^{(\ell^*+n)}_{\text{pruned}} = \mathbf{X}^{(\ell^*)}$, so the activation received by the first surviving layer differs from $\mathbf{X}^{(\ell^*+n)}$ by exactly $\mathbf{\Delta}^{(\ell^*,n)}$. All downstream layers, which were trained to process $\mathbf{X}_{\text{post}}$, instead receive $\mathbf{X}_{\text{pre}}$, and this activation mismatch propagates through the network.

Given a calibration corpus $\mathcal{D}$ of N sequences of length T (so $T_{\mathcal{D}} = NT$ tokens in total), we collect the hidden states at the two boundary layers from the original model: $\mathbf{X}_{\text{pre}}, \mathbf{X}_{\text{post}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$, where rows correspond to flattened tokens. We define the *boundary activation gap*:

$$\mathbf{\Delta} \triangleq \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}, \tag{3}$$

which is the empirical estimate of $\mathbf{\Delta}^{(\ell^*,n)}$ over $\mathcal{D}$, and quantifies the activation mismatch that any recovery method must reduce.

3.2. Ghosted Layers

Ghosted Layers is a calibration-based plug-and-play module that reduces the boundary activation mismatch via a closed-form optimal linear operator. Given a small calibration set $\mathcal{D}$, the method proceeds in three offline steps: collect the boundary activations, compute the optimal linear operator in closed form, and insert it into the pruned model.

3.2.1. Collecting boundary activations.

We run the original model $\mathcal{M}$ on $\mathcal{D}$ and capture the hidden states at the two boundary layers via forward pre-hooks. A forward pre-hook fires immediately before a layer's computation, so $\mathbf{X}_{\text{pre}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$ captures the input to the first pruned layer ℓ^*, and $\mathbf{X}_{\text{post}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$ captures the input to the first surviving layer ℓ^*+n. The boundary activation gap is then: $\mathbf{\Delta} = \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$, which equals $\mathbf{\Delta}^{(\ell^*,n)}$ from Eq. (2) evaluated over $\mathcal{D}$.

3.2.2. Closed-form optimal operator.

We seek a linear operator $\mathbf{W} \in \mathbb{R}^{C \times C}$ that, when applied to the pre-boundary state, best approximates the post-boundary state of the unpruned model:

$$\mathbf{W}^* = \arg\min_{\mathbf{W} \in \mathbb{R}^{C \times C}} \left\| \mathbf{X}_{\text{pre}} \mathbf{W} - \mathbf{X}_{\text{post}} \right\|_F^2. \tag{4}$$

Reparameterizing $\mathbf{W} = \mathbf{I} + \mathbf{M}$ and substituting into Eq. (4) yields an equivalent objective:

$$\min_{\mathbf{M} \in \mathbb{R}^{C \times C}} \left\| \mathbf{X}_{\text{pre}} \mathbf{M} - \mathbf{\Delta} \right\|_F^2. \tag{5}$$

This reparameterization transforms the alignment problem into directly regressing the boundary activation gap $\mathbf{\Delta} = \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}}$ from the pre-boundary state $\mathbf{X}_{\text{pre}}$. Rather than learning to reproduce the full post-boundary activation $\mathbf{X}_{\text{post}}$, the operator $\mathbf{M}$ only needs to capture the incremental change induced by the pruned block. This decoupling isolates the target of learning to the quantity that actually differs between the pruned and unpruned models.

The minimum-norm least-squares solution to Eq. (5) is:

$$\mathbf{M}^* = \mathbf{X}_{\text{pre}}^{\dagger} \mathbf{\Delta} = \mathbf{V} \mathbf{\Sigma}^{\dagger} \mathbf{U}^{\top} \mathbf{\Delta} \in \mathbb{R}^{C \times C}, \tag{6}$$

where $\mathbf{X}_{\text{pre}} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^{\top}$ is the thin SVD with $\sigma_1 \geq \cdots \geq \sigma_C \geq 0$, and:

$$\left(\mathbf{\Sigma}^{\dagger}\right)_{ii} = \begin{cases} \sigma_i^{-1} & \text{if } \sigma_i > \epsilon\, \sigma_1, \\ 0 & \text{otherwise,} \end{cases} \qquad \epsilon = 10^{-6}. \tag{7}$$

The threshold ϵ controls numerical stability and has no effect on the solution for well-conditioned $\mathbf{X}_{\text{pre}}$. $\mathbf{M}^*$ is a full $C \times C$ matrix with no structural constraint.

Then, the Ghosted Layers operator is $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^* \in \mathbb{R}^{C \times C}$.

3.2.3. Inserting the Ghosted Layers operator.

After removing $\mathcal{B}$ and re-indexing the surviving layers, we insert $\mathbf{W}^*$ at the output of layer $\ell^* - 1$ via a forward hook. Given a hidden state $\mathbf{x} \in \mathbb{R}^{B \times T \times C}$, the module computes:

$$\mathbf{x}_{\text{new}} = \mathbf{x}\, \mathbf{W}^* = \underbrace{\mathbf{x}}_{\text{identity}} + \underbrace{\mathbf{x}\, \mathbf{M}^*}_{\text{additive}}, \tag{8}$$

where the additive term $\mathbf{x}\,\mathbf{M}^*$ serves as the learned estimate of the boundary gap $\mathbf{\Delta}$: since $\mathbf{M}^*$ minimizes $\|\mathbf{X}_{\text{pre}}\mathbf{M} - \mathbf{\Delta}\|_F^2$, the product $\mathbf{x}\,\mathbf{M}^*$ approximates the incremental update that the pruned layers would have contributed to $\mathbf{x}$. When $\mathbf{M}^* = \mathbf{0}$, Eq. (8) reduces to $\mathbf{x}_{\text{new}} = \mathbf{x}$, recovering the pruned baseline exactly.

Practical. We compute $\mathbf{M}^*$ by solving a regularized least-squares system via `torch.linalg.solve`, using $\epsilon = 10^{-6}$. In practice, this yields the same solution as the pseudoinverse formulation, as the regularization stabilizes the system when $T_{\mathcal{D}} \gg C$. For reproducibility, we provide complete implementation details, including both SVD-based and solver-based variants, in Appendix J.

4. Unconstrained solution space analysis

We empirically analyze the properties of the unconstrained optimal operator $\mathbf{M}^*$ to validate the theoretical claims established in Section 3. All experiments in this section use n=7 pruned layers across two LLM backbones; detailed experimental settings are provided in Appendix C.

Theorem 4.1 establishes that $\mathbf{W}^*$ is the unconstrained minimizer of LinearPatch's own alignment objective, and that LinearPatch's symmetric parameterization prevents it from attaining this solution.

Theorem 4.1 (Ghosted Layers is the Unconstrained Optimum of LinearPatch)**.** *Let* $\mathbf{X}_{\text{pre}}, \mathbf{X}_{\text{post}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$ *and* $\mathbf{\Delta} = \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}}$*. Both LinearPatch and Ghosted Layers produce a repaired activation of the form* $\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{pre}}\mathbf{W}$*, but differ in the structure of* $\mathbf{W}$*:*

$$\mathbf{X}_{\text{new,LP}} = \mathbf{X}_{\text{pre}} \cdot \underbrace{\mathbf{H}\mathbf{D}\mathbf{H}^\top}_{\mathbf{W}_{\text{LP}},\ \mathbf{W}_{\text{LP}}^\top = \mathbf{W}_{\text{LP}}} \tag{9}$$

$$\mathbf{X}_{\text{new,GL}} = \mathbf{X}_{\text{pre}} \cdot \underbrace{(\mathbf{I} + \mathbf{M}^*)}_{\mathbf{W}^*,\ \mathbf{W}^* \in \mathbb{R}^{C \times C}} \tag{10}$$

where $\mathbf{M}^* = \mathbf{X}_{\text{pre}}^\dagger \mathbf{\Delta}$*, and* $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ *is the minimum-norm solution to the unconstrained alignment objective* $\min_{\mathbf{W}} \|\mathbf{X}_{\text{pre}}\mathbf{W} - \mathbf{X}_{\text{post}}\|_F^2$*, whereas* $\mathbf{W}_{\text{LP}}$ *searches only over symmetric matrices, making LinearPatch a constrained approximation to Ghosted Layers.*

Theorem 4.1 thus positions LinearPatch as a constrained special case of the activation alignment framework: both methods optimize the same alignment objective over the same functional form $\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{pre}}\mathbf{W}$, yet LinearPatch confines its search to the symmetric subspace of dimension $\frac{C(C+1)}{2}$, rendering $\mathbf{W}^*$ structurally inaccessible whenever it has a non-zero anti-symmetric component. The proof is deferred to Appendix B.1.

Constrained solution space. As established in Theorem 4.1, $\mathbf{W}^*$ is the unconstrained minimizer over all of $\mathbb{R}^{C \times C}$, whereas any symmetric operator $\mathbf{W}$ satisfies $\mathbf{W} - \mathbf{W}^\top = \mathbf{0}$ and is thus confined to the symmetric subspace. To empirically verify that $\mathbf{W}^*$ indeed lies outside this subspace, we compute $\mathbf{M}^*$ from the calibration activations $\mathbf{X}_{\text{pre}}, \mathbf{X}_{\text{post}}$ collected from the original unpruned model, and decompose $\mathbf{M}^*$ into its symmetric and anti-symmetric components:

$$\mathbf{M}^* = \underbrace{\frac{\mathbf{M}^* + (\mathbf{M}^*)^\top}{2}}_{\mathbf{M}^*_{\text{sym}}} + \underbrace{\frac{\mathbf{M}^* - (\mathbf{M}^*)^\top}{2}}_{\mathbf{M}^*_{\text{asym}}}.$$

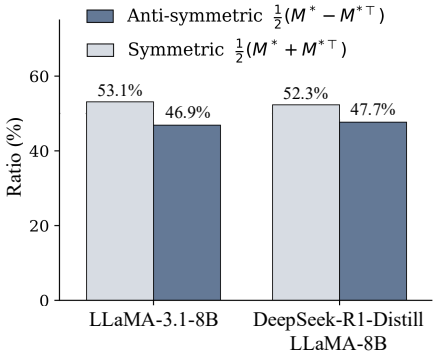

Figure 3. Frobenius norm decomposition of $\mathbf{M}^*$ into symmetric and anti-symmetric components across two LLM backbones (n = 7, LLM-Streamline). Detailed setups are in Appendix B.2

Figure 3 shows that $\mathbf{M}^*_{\text{asym}}$ is non-zero and comparable in magnitude to $\mathbf{M}^*_{\text{sym}}$ consistently across all two backbones. Since any symmetric operator satisfies $\mathbf{M}_{\text{asym}} = \mathbf{0}$ by construction, this anti-symmetric component is structurally inaccessible to LinearPatch regardless of how its diagonal $\mathbf{D}$ is chosen.

5. Experimental results

5.1. Experimental setups

Benchmarks. We evaluate the performance of *Ghosted Layers* on nine zero-shot commonsense reasoning benchmarks: `ARC-Easy` and `ARC-Challenge` (Clark et al., 2018), `HellaSwag` (Zellers et al., 2019), `WinoGrande` (Sakaguchi et al., 2019), `BoolQ` (Clark et al., 2019), `OpenbookQA` (Mihaylov et al., 2018), `RTE` (Dagan et al., 2005), `COPA` (Roemmele et al., 2011), and `RACE` (Lai et al., 2017), using the `lm-evaluation-harness` framework (Gao et al., 2024). For perplexity, we evaluate on `WikiText-2` (Merity et al., 2017), `C4` (Raffel et al., 2020), and `Penn Treebank` (Marcus et al., 1993) using non-overlapping windows of length $T = 2{,}048$.

Models. We evaluate `Ghosted Layers` on two open-source LLMs: LLaMA-3.1-8B (Grattafiori et al., 2024) and DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025). All experiments are conducted on a single NVIDIA A40 48GB

Table 1. Zero-shot accuracy (%) on commonsense QA benchmarks for 7-layer and 11-layer pruning across two LLM backbones. All methods are training-free. AVG denotes the mean accuracy across all nine tasks. L_p/L_t denotes the number of pruned layers L_p over the total number of layers L_t in the original model.

Model	L_p/L_t	Method	ARC-E	ARC-C	HellaS	WinoG	BoolQ	OBQA	RTE	CoPa	Race	AVG ↑
LLaMA-3.1-8B	0/32	Dense	81.19	53.41	78.92	73.64	82.11	44.80	69.68	87.00	39.14	67.77
	7/32	Shortened LLaMA	61.32	33.11	59.55	54.22	43.76	35.20	51.62	77.00	31.29	49.67
	7/32	LLM-Streamline	44.19	33.11	33.39	56.99	38.20	32.60	58.12	61.00	25.84	42.60
	7/32	ShortGPT	58.29	42.15	64.96	68.35	61.99	34.40	69.68	80.00	34.45	57.14
	7/32	Prune and Comp	46.25	30.89	44.22	59.12	53.91	35.00	60.29	67.00	28.61	47.25
	7/32	ReplaceMe (Ls)	64.90	43.52	63.80	71.67	69.60	37.80	71.48	77.00	37.32	59.68
	7/32	ReplaceMe (Cos)	57.03	37.29	52.14	64.25	39.36	35.80	62.82	68.00	31.10	49.75
	7/32	Linear Patch (Diag)	48.36	32.68	47.36	61.56	63.70	35.00	68.23	68.00	29.28	50.46
	7/32	Linear Patch (Rotate)	57.62	37.29	54.91	65.43	60.49	35.80	69.68	69.00	29.19	53.27
	7/32	Ghost Layer (Ours)	68.01	43.00	66.60	71.59	71.31	37.20	68.59	76.00	37.80	60.01
	11/32	Shortened LLaMA	38.54	30.38	29.05	50.91	59.80	28.20	50.26	62.00	29.38	42.05
	11/32	LLM-Streamline	39.69	30.29	31.49	56.12	55.11	30.40	69.68	61.00	28.33	44.68
	11/32	ShortGPT	39.69	30.29	31.49	56.12	55.11	30.40	69.68	61.00	28.33	44.68
	11/32	Prune and Comp	38.51	27.30	40.50	53.28	62.08	29.40	53.79	57.00	25.17	43.00
	11/32	ReplaceMe (Ls)	44.53	34.04	47.23	67.40	73.55	29.80	70.76	67.00	29.95	51.58
	11/32	ReplaceMe (Cos)	43.35	32.59	37.02	56.75	58.13	29.60	68.23	62.00	31.96	46.63
	11/32	Linear Patch (Diag)	50.67	36.60	48.55	62.51	71.31	31.00	71.84	68.00	31.00	52.39
	11/32	Linear Patch (Rotate)	50.93	34.98	47.30	62.19	74.65	30.40	71.48	68.00	31.58	52.39
	11/32	Ghost Layer (Ours)	50.08	34.64	52.73	69.30	72.63	31.80	69.68	72.00	32.44	53.92
DeepSeek-R1-Distill-LLaMA-8B	0/32	Dense	65.87	42.41	74.35	67.80	82.91	41.20	69.68	89.00	41.63	63.87
	7/32	Shortened LLaMA	45.71	29.69	55.66	57.70	64.65	30.40	50.54	76.00	30.62	49.00
	7/32	LLM-Streamline	49.92	35.24	46.33	61.56	51.99	32.40	57.40	70.00	27.27	48.01
	7/32	ShortGPT	49.12	37.03	55.95	63.06	77.09	34.00	74.73	71.00	33.21	55.02
	7/32	Prune and Comp	44.99	31.06	43.44	56.12	53.12	31.80	60.65	64.00	26.79	45.77
	7/32	ReplaceMe (Ls)	55.35	37.03	59.97	66.46	69.08	35.20	75.09	80.00	37.80	57.33
	7/32	ReplaceMe (Cos)	51.09	36.09	53.48	64.17	58.01	36.00	58.84	74.00	33.49	51.69
	7/32	Linear Patch (Diag)	51.09	35.32	49.59	62.27	62.78	34.60	66.43	73.00	30.05	51.68
	7/32	Linear Patch (Rotate)	54.08	36.18	53.29	64.25	72.97	33.80	62.82	72.00	32.92	53.59
	7/32	Ghost Layer (Ours)	55.81	37.03	61.30	67.25	68.47	37.00	72.92	81.00	39.43	57.80
	11/32	Shortened LLaMA	41.75	28.84	42.84	54.30	55.23	27.80	53.79	70.00	26.99	44.62
	11/32	LLM-Streamline	37.08	31.66	38.72	55.17	75.20	27.40	64.26	63.00	26.22	46.52
	11/32	ShortGPT	37.08	31.66	38.72	55.17	75.20	27.40	64.26	63.00	26.22	46.52
	11/32	Prune and Comp	36.45	29.10	36.32	50.99	64.62	27.40	60.29	56.00	24.40	42.84
	11/32	ReplaceMe (Ls)	40.19	30.63	44.70	61.40	66.45	32.00	74.73	65.00	33.78	49.88
	11/32	ReplaceMe (Cos)	38.05	30.55	40.74	54.93	77.52	28.60	67.51	65.00	30.81	48.19
	11/32	Linear Patch (Diag)	45.58	34.90	43.97	58.33	77.46	31.60	68.23	64.00	29.57	50.40
	11/32	Linear Patch (Rotate)	43.69	32.76	44.22	59.19	77.34	30.40	70.04	65.00	30.72	50.37
	11/32	Ghost Layer (Ours)	42.72	32.59	48.09	64.33	75.41	31.80	74.01	66.00	34.07	52.11

GPU.

Layer pruning and recovery methods. We use three layer selection criteria: `LLM-Streamline` (Chen et al., 2025a), `ShortGPT` (Men et al., 2025), and `Shortened LLaMA` (Kim et al., 2024), all using their official implementations. For recovery, we compare the following training-free methods: `Prune&Comp` (Chen et al., 2025b), `LinearPatch (Diag/Rotate)` (Chen et al., 2026), `ReplaceMe (LS/Cos)` (Shopkhoev et al., 2026), and `Ghosted Layers` (ours). All baselines use official implementations, except `Prune&Comp`, which we reimplement from the paper as no public code is available. Our implementation builds on the `LinearPatch` codebase. For layer selection and operator estimation, we use 128 randomly sampled sequences of length $T = 2{,}048$ from the C4 training split (Raffel et al., 2020).

5.2. Numerical results

Results on QA Benchmarks. Table 1 reports zero-shot accuracy on nine commonsense QA benchmarks for 7-layer and 11-layer pruning across two LLM backbones, with LLM-Streamline as the pruning criterion. Across both pruning ratios and both backbones, `Ghosted Layers` attains the highest or competitive average accuracy among training-free recovery methods. Results on additional backbones (LLaMA-2-7B (Touvron et al., 2023b), OLMo-2-7B (Walsh et al., 2025), Qwen3-14B (Yang et al., 2025)) are provided in Appendix G.

Results on PPL Benchmarks. Table 2 reports perplexity on WikiText-2, C4, and Penn Treebank across two backbones and three pruning criteria. Among these, LLM-Streamline removes a contiguous block of layers, whereas ShortGPT and Shortened LLaMA prune layers non-contiguously. Some pruned models exhibit markedly elevated perplexity, a behavior consistent with observations reported in prior work (Chen et al., 2026). Across all three pruning criteria, `Ghosted Layers` achieves the lowest average perplexity in most combinations, and the margin widens under the more aggressive 11-layer setting.

Efficiency Comparison. Table 3 reports GPU memory, prefill latency, accuracy, and perplexity for LLaMA-3.1-8B under 11/32 layer pruning (sequence length 2,048, batch size 1). Latency is measured as the mean prefill time over 10 runs with 3 warmup iterations, and GPU memory is reported as the peak activated tensor footprint. `LinearPatch` and `Ghosted Layers` incur identical cost as a single $C \times C$

Table 2. Comparison on PPL benchmark with training-free methods over two LLMs. L_p/L_t denotes the number of pruned layers L_p over the total number of layers L_t in the original model.

Model	Method	7-layer WIKI↓	C4↓	PTB↓	PPL AVG↓	ACC AVG↑	11-Layer WIKI↓	C4↓	PTB↓	PPL AVG↓	ACC AVG↑
LLaMA-3.1-8B	Dense	6.24	8.68	10.58	8.50	67.77	6.24	8.68	10.58	8.50	67.77
	Shortened LLaMA	14.54	18.44	22.82	18.60	50.16	54.14	47.15	152.18	84.49	42.05
	+ Prune and Comp	29.41	40.46	44.26	38.04	42.60	182.71	119.57	253.54	185.27	41.22
	+ ReplaceMe (Ls)	70.03	55.41	134.37	86.60	43.84	1176.61	444.62	1827.31	1149.51	38.85
	+ ReplaceMe (Cos)	14.40	18.35	22.68	18.48	49.79	46.80	38.71	58.73	48.08	42.05
	+ Linear Patch (Diag)	12.45	17.50	20.35	16.77	41.38	224.29	136.05	314.63	224.99	40.84
	+ Linear Patch (Rotate)	12.46	17.17	20.27	16.63	43.75	232.96	180.62	343.98	252.52	41.41
	+ Ghost Layer (Ours)	10.79	15.26	18.26	14.77	53.55	78.81	64.56	196.16	113.18	42.31
	LLM-Streamline	2301.46	1173.53	3720.23	2398.41	42.60	4799.22	6510.32	6173.75	5827.76	44.68
	+ Prune and Comp	157.93	191.65	207.59	185.72	47.25	543.58	404.81	841.99	596.79	43.00
	+ ReplaceMe (Ls)	29.68	26.27	51.11	35.69	59.68	133.21	78.68	400.33	204.07	51.58
	+ ReplaceMe (Cos)	147.97	131.94	195.69	158.53	49.75	1469.20	1487.28	1817.19	1591.22	46.63
	+ Linear Patch (Diag)	110.93	134.68	135.50	127.04	50.46	224.32	195.13	391.71	270.39	52.39
	+ Linear Patch (Rotate)	57.33	73.48	67.38	66.06	53.27	232.96	180.62	343.98	252.52	52.39
	+ Ghost Layer (Ours)	21.35	21.52	40.56	27.81	60.01	54.98	41.31	125.73	74.01	53.92
	ShortGPT	63.42	69.64	68.58	67.21	57.10	3799.22	2510.32	4173.75	3494.43	44.71
	+ Prune and Comp	29.41	40.46	44.26	38.04	55.91	669.00	505.01	998.22	724.08	40.85
	+ ReplaceMe (Ls)	519.94	318.59	137.12	325.22	36.47	1253.00	592.00	286.00	710.33	37.48
	+ ReplaceMe (Cos)	64.41	67.57	103.02	78.33	56.43	423.14	538.34	425.84	462.44	39.55
	+ Linear Patch (Diag)	27.03	36.95	34.14	32.71	59.12	208.68	182.72	372.63	254.68	52.58
	+ Linear Patch (Rotate)	33.16	39.51	39.27	37.31	58.90	431.44	351.86	741.12	508.14	50.93
	+ Ghost Layer (Ours)	15.72	19.73	27.01	20.82	60.45	61.14	50.74	116.81	76.23	54.02
DeepSeek-R1-Distill-LLaMA-8B	Dense	13.13	19.46	22.28	18.29	63.87	13.13	19.46	22.28	18.29	63.87
	Shortened LLaMA	29.26	37.19	45.26	37.24	49.00	94.06	87.80	127.20	103.02	44.62
	+ Prune and Comp	26.82	34.22	42.28	34.44	49.29	387.30	198.67	399.84	328.60	40.08
	+ ReplaceMe (Ls)	54.69	58.63	78.63	63.98	45.94	947.48	420.19	895.78	754.48	38.16
	+ ReplaceMe (Cos)	27.60	35.12	42.89	35.20	49.75	95.91	84.17	134.11	104.73	45.21
	+ Linear Patch (Diag)	23.09	30.32	35.74	29.72	51.21	45.64	50.10	66.09	53.94	46.06
	+ Linear Patch (Rotate)	23.66	30.84	36.32	30.27	51.69	45.08	48.64	63.73	52.48	45.90
	+ Ghost Layer (Ours)	20.34	26.77	31.72	26.28	52.82	33.66	39.54	46.21	39.80	47.25
	LLM-Streamline	3083.95	1291.64	2985.68	2453.76	48.01	5094.57	4171.75	4114.49	4460.27	46.52
	+ Prune and Comp	805.43	521.87	1243.46	856.92	45.77	3061.56	1009.17	4305.52	2792.08	42.84
	+ ReplaceMe (Ls)	65.41	48.49	101.53	71.81	57.33	275.22	138.16	485.00	299.46	49.88
	+ ReplaceMe (Cos)	441.74	312.40	578.42	444.19	51.69	4059.05	2408.55	8476.96	4981.52	48.19
	+ Linear Patch (Diag)	476.26	341.89	596.51	471.55	51.68	1512.90	585.54	1998.35	1365.60	50.40
	+ Linear Patch (Rotate)	319.22	212.18	459.40	330.27	53.59	2070.12	536.66	2752.07	1786.28	50.37
	+ Ghost Layer (Ours)	45.27	38.69	59.37	47.78	57.80	115.18	75.07	181.76	124.00	52.11
	ShortGPT	343.44	157.21	906.19	468.95	55.05	4911.51	4261.16	3994.34	4389.00	46.54
	+ Prune and Comp	131.62	86.37	277.45	165.15	51.13	2901.15	1443.53	3941.85	2762.18	42.68
	+ ReplaceMe (Ls)	1225.19	793.62	3787.38	1935.40	42.78	4096.38	5405.75	4597.50	4699.88	38.89
	+ ReplaceMe (Cos)	579.23	171.66	3628.31	1459.73	54.84	4505.67	4341.19	3178.56	4008.47	44.52
	+ Linear Patch (Diag)	88.29	69.16	160.98	106.14	57.11	1502.95	577.91	1994.45	1358.44	50.37
	+ Linear Patch (Rotate)	110.34	66.54	211.78	129.55	56.89	3654.27	847.01	4649.31	3050.20	49.45
	+ Ghost Layer (Ours)	30.30	33.71	41.29	35.10	57.87	131.41	81.61	201.41	138.14	52.31

Table 3. Inference cost, accuracy, and perplexity on LLaMA-3.1-8B under 11/32 layer pruning. GPU memory is reported as the peak activated tensor footprint during the forward pass measured via `torch.cuda.max_memory_allocated()`, normalized to the dense baseline. Latency is the mean prefill time at sequence length 2,048 over 10 runs with 3 warmup iterations on the same input.

Method	L_p/L_t	GPU (%)	Latency (ms)	Speedup↑	ACC AVG↑	PPL AVG↓
Dense	0/32	100.0	362.6	1.00×	67.77	8.50
LLM-Streamline	11/32	71.1	244.4	1.48×	44.68	5827.76
+ Prune and Comp	11/32	71.1	244.9	1.48×	43.00	596.79
+ ReplaceMe (LS)	11/32	71.1	244.8	1.48×	51.58	204.07
+ ReplaceMe (Cos)	11/32	71.1	245.2	1.48×	46.63	1591.22
+ Linear Patch (D)	11/32	71.5	248.1	1.46×	52.39	270.39
+ Linear Patch (R)	11/32	71.5	248.1	1.46×	52.39	252.52
+ Ghost Layer (Ours)	11/32	71.5	247.6	1.46×	53.92	74.01

operation. Under this matched cost, `Ghosted Layers` achieves higher accuracy and substantially lower perplexity than `LinearPatch`.

6. Conclusion

We presented Ghosted Layers, a training-free recovery module for layer-pruned large language models that addresses boundary activation mismatch. Our approach derives a closed-form optimal linear operator from a small calibration set to reconstruct the activation discrepancy introduced by pruning. Unlike prior methods that operate over constrained operator classes, Ghosted Layers solves the alignment problem in the unconstrained space of linear operators. Across multiple pruning strategies and LLM backbones, Ghosted Layers consistently improves accuracy and perplexity over prior training-free recovery methods at matched inference cost. These results suggest that solving boundary activation alignment at its unconstrained optimum is sufficient to substantially recover the performance of layer-pruned models without additional inference overhead.

Limitation. Ghosted Layers requires a small unlabeled calibration set to collect boundary activations from the unpruned model and compute the recovery operator. Similar calibration requirements are shared by prior post-pruning recovery methods such as LinearPatch and ReplaceMe. Computing the recovery operator further requires a one-time offline least-squares solve, which does not affect inference cost.

References

An, Y., Zhao, X., Yu, T., Tang, M., and Wang, J. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28960. URL https://doi.org/10.1609/aaai.v38i10.28960.

Ashkboos, S., Croci, M. L., do Nascimento, M. G., Hoefler, T., and Hensman, J. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chen, X., Hu, Y., Zhang, J., Wang, Y., Li, C., and Chen, H. Streamlining redundant layers to compress large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=IC5RJvRoMp.

Chen, X., Zhang, H., Zeng, F., Wei, Y., Wang, Y., Ling, X., Li, G., and Yuan, C. Prune&comp: Free lunch for layer-pruned LLMs via iterative pruning with magnitude compensation. *arXiv preprint arXiv:2507.18212*, 2025b.

Chen, X., Bai, H., Yuan, T., Liu, R., Zhao, K., Yu, X., Hou, L., Guan, T., He, Y., and Yuan, C. A simple linear patch revives layer-pruned large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=AwsiYZ2ets.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dagan, I., Glickman, O., and Magnini, B. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pp. 177–190. Springer, 2005.

Frantar, E. and Alistarh, D. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

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

Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P., and Roberts, D. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

Guo, D., Yang, D., Zhang, H., et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Higham, N. J. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2002.

Kim, B.-K., Kim, G., Kim, T.-H., Castells, T., Choi, S., Shin, J., and Song, H.-K. Shortened LLaMA: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*, 2024.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*,

pp. 785–794, 2017. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Ma, X., Fang, G., and Wang, X. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=J8Ajf9WfXP.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Men, X., Xu, M., Zhang, Q., Yuan, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, July 2025. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Muralidharan, S., Turuvekere Sreenivas, S., Joshi, R., Chochowski, M., Patwary, M., Shoeybi, M., Catanzaro, B., Kautz, J., and Molchanov, P. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 41076–41102, 2024. doi: 10.52202/079017-1299.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (1), 2020. ISSN 1532-4435.

Roemmele, M., Bejan, C. A., and Gordon, A. S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Shopkhoev, D., Ali, A., Zhussip, M., Malykh, V., Lefkimmiatis, S., Komodakis, N., and Zagoruyko, S. Replaceme: Network simplification via depth pruning and transformer block linearization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=zEj1FSYCRn.

Song, J., Oh, K., Kim, T., Kim, H., Kim, Y., and Kim, J.-J. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Walsh, E. P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N.,

Ettinger, A., Guerquin, M., Heineman, D., Ivison, H., Koh, P. W., Liu, J., Malik, S., Merrill, W., Miranda, L. J. V., Morrison, J., Murray, T., Nam, C., Poznanski, J., Pyatkin, V., Rangapur, A., Schmitz, M., Skjonsberg, S., Wadden, D., Wilhelm, C., Wilson, M., Zettlemoyer, L., Farhadi, A., Smith, N. A., and Hajishirzi, H. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

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

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Appendix: Ghosted Layers

This appendix provides supplementary materials that complement the main paper. It includes the proof of our main theoretical result, detailed experimental settings, and additional quantitative results across calibration sizes, model architectures, fine-tuning protocols, and calibration datasets. The appendix is organized as follows:

- **Appendix** A: Extended related works
- **Appendix** B: Proof for Theorem 4.1 and Constrained Solution space analysis
- **Appendix** C: Experimental setups
- **Appendix** D: Extended results (calibration size)
- **Appendix** G: Extended results (Additional LLM architectures)
- **Appendix** E: Extended results (fine-tuning)
- **Appendix** F: Extended results (Alternative calibration datasets)
- **Appendix** H: Extended results (Effect of pruning depth on accuracy)
- **Appendix** I: Efficiency measurement
- **Appendix** J: Closed-form solution computation

A. Extended related works

Structured pruning of LLMs. Structured pruning reduces model size by removing groups of parameters while preserving dense computation, enabling deployment without specialized kernels or sparse runtime support. Width pruning removes attention heads, MLP neurons, or hidden dimensions using importance scores derived from weights (Ma et al., 2023), activations (An et al., 2024; Ashkboos et al., 2024), or gradients (Xia et al., 2023), but often introduces architectural irregularities and requires retraining or distillation to recover performance (Muralidharan et al., 2024). In contrast, depth pruning removes entire Transformer blocks while preserving the original architecture and requiring no specialized hardware support.

Layer pruning. Various metrics have been proposed to identify redundant layers. ShortGPT (Men et al., 2025) uses a Block Influence (BI) score based on cosine similarity between input and output activations for one-shot pruning. SLEB (Song et al., 2024) iteratively removes layers that minimally increase perplexity on a calibration set. Shortened LLaMA (Kim et al., 2024) evaluates each layer via its perplexity impact under removal. LLM-Streamline (Chen et al., 2025a) instead removes a *contiguous* block of layers with high boundary activation similarity. These methods primarily focus on selecting *which* layers to remove, leaving boundary mismatch to be handled separately.

Post-pruning recovery. Prune&Comp (Chen et al., 2025b) rescales the surviving boundary weights with per-channel scalars, capturing only channel-wise magnitude changes and no cross-channel interaction. `ReplaceMe` (Shopkhoev et al., 2026) instead approximates the pruned block's computation with a linear map of the boundary MLP output, absorbed into the surviving MLP weights, so the repair acts on the block's computation rather than on the boundary hidden state itself. `LinearPatch` (Chen et al., 2026) directly targets the boundary activation mismatch with a symmetric linear operator parameterized as a Hadamard-rotated diagonal scaling, confining the search to a strict subspace of $\mathbb{R}^{C\times C}$. Our `Ghosted Layers` shares the goal of reducing the boundary activation mismatch, but operates over the unconstrained space of linear maps and solves the resulting alignment problem in closed form, recovering structure that the channel-wise, block-level, and symmetric parameterizations above cannot express.

B. Constrained solution space analysis

B.1. Proof of Theorem B.1

This appendix provides the proof of Theorem 4.1, which establishes that $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ is the minimum-norm solution to the unconstrained activation alignment objective, and that `LinearPatch`'s symmetric parameterization constitutes a strict subspace restriction of this solution. For completeness, we restate the theorem below before presenting its proof.

Theorem B.1 (Ghosted Layers is the Unconstrained Optimum of LinearPatch)**.** *Let* $\mathbf{X}_{\text{pre}}, \mathbf{X}_{\text{post}} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$ *and* $\mathbf{\Delta} = \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}}$*. Both LinearPatch and Ghosted Layers produce a repaired activation of the form* $\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{pre}}\mathbf{W}$*, but differ in the structure of* $\mathbf{W}$*:*

$$\mathbf{X}_{\text{new,LP}} = \mathbf{X}_{\text{pre}} \cdot \underbrace{\mathbf{H}\mathbf{D}\mathbf{H}^\top}_{\mathbf{W}_{\text{LP}},\ \mathbf{W}_{\text{LP}}^\top = \mathbf{W}_{\text{LP}}} \qquad \mathbf{X}_{\text{new,GL}} = \mathbf{X}_{\text{pre}} \cdot \underbrace{(\mathbf{I} + \mathbf{M}^*)}_{\mathbf{W}^*,\ \mathbf{W}^* \in \mathbb{R}^{C \times C}} \tag{A1}$$

where $\mathbf{M}^* = \mathbf{X}_{\text{pre}}^\dagger \mathbf{\Delta}$*, and* $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ *is the minimum-norm solution to the unconstrained alignment objective* $\min_{\mathbf{W}} \|\mathbf{X}_{\text{pre}}\mathbf{W} - \mathbf{X}_{\text{post}}\|_F^2$*, whereas* $\mathbf{W}_{\text{LP}}$ *searches only over symmetric matrices, making LinearPatch a constrained approximation to Ghosted Layers.*

Proof. Both methods produce $\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{pre}}\mathbf{W}$.

Substituting $\mathbf{W} = \mathbf{I} + \mathbf{M}$ into the alignment objective:

$$\begin{aligned} \|\mathbf{X}_{\text{pre}}\mathbf{W} - \mathbf{X}_{\text{post}}\|_F^2 &= \|\mathbf{X}_{\text{pre}}(\mathbf{I} + \mathbf{M}) - \mathbf{X}_{\text{post}}\|_F^2 \\ &= \|\mathbf{X}_{\text{pre}}\mathbf{M} - \underbrace{(\mathbf{X}_{\text{post}} - \mathbf{X}_{\text{pre}})}_{\mathbf{\Delta}}\|_F^2 \\ &= \|\mathbf{X}_{\text{pre}}\mathbf{M} - \mathbf{\Delta}\|_F^2. \end{aligned} \tag{A2}$$

Hence minimizing over $\mathbf{W}$ is equivalent to $\min_{\mathbf{M}} \|\mathbf{X}_{\text{pre}}\mathbf{M} - \mathbf{\Delta}\|_F^2$.

Since the Frobenius norm decouples across columns, it suffices to solve independently for each column $j = 1, \ldots, C$:

$$\min_{\mathbf{m}_j \in \mathbb{R}^C} \|\mathbf{X}_{\text{pre}}\mathbf{m}_j - \mathbf{\Delta}_{:,j}\|_2^2. \tag{A3}$$

Taking the gradient with respect to $\mathbf{m}_j$ and setting it to zero:

$$\nabla_{\mathbf{m}_j} \|\mathbf{X}_{\text{pre}}\mathbf{m}_j - \mathbf{\Delta}_{:,j}\|_2^2 = 2\mathbf{X}_{\text{pre}}^\top(\mathbf{X}_{\text{pre}}\mathbf{m}_j - \mathbf{\Delta}_{:,j}) = \mathbf{0}, \tag{A4}$$

which yields the normal equations:

$$\mathbf{X}_{\text{pre}}^\top \mathbf{X}_{\text{pre}}\, \mathbf{m}_j = \mathbf{X}_{\text{pre}}^\top \mathbf{\Delta}_{:,j}. \tag{A5}$$

When $\mathbf{X}_{\text{pre}}$ has full column rank, $\mathbf{X}_{\text{pre}}^\top \mathbf{X}_{\text{pre}}$ is invertible. To handle the general rank-deficient case, let $\mathbf{X}_{\text{pre}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the thin SVD with $\mathbf{U} \in \mathbb{R}^{T_{\mathcal{D}} \times r}$, $\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_r)$, $\mathbf{V} \in \mathbb{R}^{C \times r}$, and $r = \text{rank}(\mathbf{X}_{\text{pre}})$. The minimum-norm least-squares solution is then:

$$\mathbf{m}_j^* = \left(\mathbf{X}_{\text{pre}}^\top \mathbf{X}_{\text{pre}}\right)^{-1} \mathbf{X}_{\text{pre}}^\top \mathbf{\Delta}_{:,j} \tag{A6}$$

$$= \mathbf{V}\mathbf{\Sigma}^\dagger \mathbf{U}^\top \mathbf{\Delta}_{:,j} \tag{A7}$$

$$= \mathbf{X}_{\text{pre}}^\dagger \mathbf{\Delta}_{:,j}, \tag{A8}$$

where the Moore–Penrose pseudoinverse $\mathbf{X}_{\text{pre}}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^\top$ with:

$$\left(\mathbf{\Sigma}^\dagger\right)_{ii} = \begin{cases} 1/\sigma_i & \text{if } \sigma_i > \epsilon\,\sigma_1, \\ 0 & \text{otherwise,} \end{cases} \qquad \epsilon = 10^{-6}. \tag{A9}$$

Stacking the per-column solutions over all $j = 1, \ldots, C$:

$$\mathbf{M}^* = \mathbf{X}_{\text{pre}}^\dagger \mathbf{\Delta} = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^\top\mathbf{\Delta}, \qquad \mathbf{W}^* = \mathbf{I} + \mathbf{M}^*. \tag{A10}$$

LinearPatch parameterizes $\mathbf{W}_{\text{LP}} = \mathbf{H}\mathbf{D}\mathbf{H}^\top$, where $\mathbf{H} \in \mathbb{R}^{C \times C}$ is the Walsh–Hadamard matrix satisfying $\mathbf{H}^\top = \mathbf{H}^{-1}$, and $\mathbf{D} = \text{diag}(d_1, \ldots, d_C)$ is a diagonal matrix.

We verify symmetry by direct computation:

$$\begin{aligned}\mathbf{W}_{\mathrm{LP}}^{\top} &= \left(\mathbf{H}\mathbf{D}\mathbf{H}^{\top}\right)^{\top} \\ &= \left(\mathbf{H}^{\top}\right)^{\top}\mathbf{D}^{\top}\mathbf{H}^{\top} \\ &= \mathbf{H}\mathbf{D}^{\top}\mathbf{H}^{\top}. \end{aligned} \tag{A11}$$

Since $\mathbf{D}$ is diagonal, $\mathbf{D}^{\top} = \mathbf{D}$, and therefore:

$$\mathbf{W}_{\mathrm{LP}}^{\top} = \mathbf{H}\mathbf{D}\mathbf{H}^{\top} = \mathbf{W}_{\mathrm{LP}}. \tag{A12}$$

Hence $\mathbf{W}_{\mathrm{LP}}$ is symmetric for any choice of $\mathbf{D}$, and lies in the space of $C \times C$ symmetric matrices, a subspace of dimension $\frac{C(C+1)}{2}$, strictly smaller than $C^2 = \dim(\mathbb{R}^{C\times C})$. Therefore, $\mathbf{W}_{\mathrm{LP}}$ cannot attain $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ whenever $\mathbf{W}^*$ has a non-zero anti-symmetric component, i.e., whenever $\mathbf{W}^* \neq (\mathbf{W}^*)^{\top}$. □

B.2. Computing the Symmetric and Anti-symmetric Decomposition

This section details the procedure used to compute the symmetric and anti-symmetric components of $\mathbf{M}^*$ reported in Figure 3 in Section 4.

B.2.1. Experimental Setups

Models. We evaluate three open-source LLM backbones: LLaMA-3-8B (Grattafiori et al., 2024), LLaMA-3.1-8B (Grattafiori et al., 2024), and DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025). All three models share the same hidden dimension C=4,096.

Calibration data. We use 128 sequences of length T=2,048 sampled from the C4 training split (Raffel et al., 2020), following LinearPatch (Chen et al., 2026). For the symmetry decomposition, we process 32 batches of these sequences to form the boundary activation matrices, yielding $T_{\mathcal{D}} = 32 \times 2{,}048 = 65{,}536$ tokens per backbone.

Pruning criterion. All backbones use the LLM-Streamline criterion (Chen et al., 2025a), which selects the contiguous block $\mathcal{B} = \{\ell^*, \ldots, \ell^* + n - 1\}$ of n=7 layers whose boundary activations exhibit the highest cosine similarity. The same 128 calibration sequences are reused for block selection and for operator computation.

B.2.2. Procedure

Step 1: Layer selection. We select the pruning block $\mathcal{B}$ using the criterion described above. For each backbone, this yields a specific start index ℓ^* and end index $\ell^* + n$ determined by the cosine-similarity ranking of boundary activations.

Step 2: Boundary activation capture. With the unpruned model $\mathcal{M}$ in evaluation mode and `use_cache=False`, we register a forward pre-hook on layer ℓ^* to capture its input $\mathbf{X}_{\mathrm{pre}}$, and a forward pre-hook on layer $\ell^* + n$ to capture its input $\mathbf{X}_{\mathrm{post}}$. A forward pre-hook fires immediately before a layer's computation, so the captured tensors correspond exactly to the boundary activations defined in Eq. 3. Hooks are detached from the computation graph and stored on CPU. The collected tensors have shape $\mathbb{R}^{T_{\mathcal{D}}\times C}$.

Step 3: Solving for $\mathbf{M}^*$. We promote $\mathbf{X}_{\mathrm{pre}}, \mathbf{X}_{\mathrm{post}}$ to `float64` and form $\mathbf{\Delta} = \mathbf{X}_{\mathrm{post}} - \mathbf{X}_{\mathrm{pre}}$. We then solve the regularized normal equations

$$\left(\mathbf{X}_{\mathrm{pre}}^{\top}\mathbf{X}_{\mathrm{pre}} + \epsilon\mathbf{I}\right)\mathbf{M}^* = \mathbf{X}_{\mathrm{pre}}^{\top}\mathbf{\Delta} \tag{A13}$$

via `torch.linalg.solve`, which internally uses an LU factorization. This is mathematically equivalent to $\mathbf{M}^* = \mathbf{X}_{\mathrm{pre}}^{\dagger}\mathbf{\Delta}$ when $\mathbf{X}_{\mathrm{pre}}$ has full column rank (Eq. 6), which holds generically whenever $T_{\mathcal{D}} \gg C$.

Step 4: Decomposition and reporting. We decompose the resulting $\mathbf{M}^* \in \mathbb{R}^{C\times C}$ into its symmetric and anti-symmetric parts using Eq. 11, compute the Frobenius norms $\|\mathbf{M}^*\|_F$, $\|\mathbf{M}^*_{\mathrm{sym}}\|_F$, and $\|\mathbf{M}^*_{\mathrm{asym}}\|_F$, and report the ratios $\|\mathbf{M}^*_{\mathrm{sym}}\|_F/\|\mathbf{M}^*\|_F$ and $\|\mathbf{M}^*_{\mathrm{asym}}\|_F/\|\mathbf{M}^*\|_F$ in Figure 3. The procedure is identical across all three backbones; no model-specific tuning is performed.

C. Experimental setups

C.1. Details on pruned models

All experiments are conducted on officially released LLM checkpoints obtained from Hugging Face, summarized in Table A1.

Table A1. Hugging Face sources for the LLM checkpoints used in our experiments.

Model	Download link
LLaMA-2-7B	https://huggingface.co/meta-llama/Llama-2-7b-hf
LLaMA-3-8B	https://huggingface.co/meta-llama/Meta-Llama-3-8B
LLaMA-3.1-8B	https://huggingface.co/meta-llama/Llama-3.1-8B
OLMo-2-7B	https://huggingface.co/allenai/OLMo-2-1124-7B
Qwen-3-14B	https://huggingface.co/Qwen/Qwen3-14B
DeepSeek-R1-Distill-Llama-8B	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B

C.2. Details on pruning and recovery methods

This section describes the pruning criteria and recovery methods used as baselines in our experiments. All baselines are reproduced from their official implementations to ensure a fair comparison. Table A2 lists the official repositories for each method.

Table A2. Official repositories of the pruning criteria and recovery methods used in our experiments.

Category	Method	Official repository
Pruning criterion	ShortGPT (Men et al., 2025)	https://github.com/sramshetty/ShortGPT
	Shortened LLaMA (Kim et al., 2024)	https://github.com/Nota-NetsPresso/shortened-llm
	LLM-Streamline (Chen et al., 2025a)	https://github.com/ruckbreasoning/llm-streamline
Recovery method	Prune&Comp (Chen et al., 2025b)	N/A
	ReplaceMe (Shopkhoev et al., 2026)	https://github.com/mts-ai/ReplaceMe
	LinearPatch (Chen et al., 2026)	https://github.com/chenxinrui-tsinghua/LinearPatch
	`Ghosted Layers` (Ours)	Released upon acceptance

C.2.1. Pruning criteria

ShortGPT (Men et al., 2025). ShortGPT assigns each layer a Block Influence (BI) score defined as one minus the cosine similarity between its input and output hidden states, averaged over a calibration set. Layers with the lowest BI scores are removed in a one-shot manner. We reproduce ShortGPT using its official implementation, and BI scores are computed on the same 128 C4 sequences used throughout the paper.

Shortened LLaMA (Kim et al., 2024). Shortened LLaMA evaluates each layer's contribution by the perplexity degradation incurred when that layer is removed from the original model, and prunes the layers with the smallest perplexity impact. We use the official implementation and the released pruned-layer indices where available.

LLM-Streamline (Chen et al., 2025a). LLM-Streamline selects a single *contiguous* block of layers to remove, choosing the block $\mathcal{B} = \{\ell^*, \ldots, \ell^* + n - 1\}$ whose boundary activations exhibit the highest cosine similarity. This minimizes the representation change across the pruned region. We reproduce LLM-Streamline using its official implementation and adopt it as the default pruning criterion in our main experiments unless otherwise specified.

C.2.2. Recovery methods

Prune&Comp (Chen et al., 2025b). Prune&Comp rescales the surviving boundary weights with per-channel scalar factors estimated from the calibration set, compensating for magnitude shifts without introducing any additional parameters or cross-channel interaction. As no official implementation is publicly available at the time of submission, we reimplement Prune&Comp from the description in the original paper, using the same 128 C4 calibration sequences as all other methods to ensure a fair comparison.

ReplaceMe (Shopkhoev et al., 2026). ReplaceMe approximates the computation of the pruned block by a linear transformation applied to the boundary block's MLP output, and absorbs this transformation into the surviving MLP weights. We reproduce ReplaceMe using its official implementation and include two variants:

- **ReplaceMe (LS)**, which estimates the linear map via least squares
- **ReplaceMe (Cos)**, which optimizes a cosine-distance objective with Adam for 10 epochs using the default hyperparameters of the official implementation.

LinearPatch (Chen et al., 2026). LinearPatch inserts a single matrix multiplication at the pruning boundary, parameterized as $\mathbf{W}_{\mathrm{LP}} = \mathbf{H}\mathbf{D}\mathbf{H}^{\top}$, where $\mathbf{H}$ is the Walsh–Hadamard matrix and $\mathbf{D}$ is a diagonal scaling matrix. This parameterization is real symmetric by construction. We reproduce LinearPatch using its official implementation and include two variants:

- **LinearPatch (Diag)**, which applies only channel-wise scaling,
- **LinearPatch (Rotate)**, which additionally applies the Hadamard rotation.

`Ghosted Layers` (Ours). `Ghosted Layers` inserts an unconstrained linear operator $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ at the pruning boundary, where $\mathbf{M}^* = \mathbf{X}_{\mathrm{pre}}^{\dagger}\mathbf{\Delta}$ is the closed-form minimum-norm solution to the alignment objective defined in Section 3. Our implementation builds on the official LinearPatch codebase and uses the same calibration set (128 sequences from C4, T=2,048) across all experiments.

C.2.3. Details of evaluation benchmarks

We assess model quality along two axes: language modeling perplexity and downstream task accuracy.

Perplexity (PPL). We report perplexity on three standard language modeling corpora: WikiText-2 (Merity et al., 2017), C4 (Raffel et al., 2020), and Penn Treebank (PTB) (Marcus et al., 1993). Perplexity is computed on non-overlapping windows of length T=2,048, consistent with the calibration sequence length.

Commonsense QA. For downstream accuracy, we evaluate on nine zero-shot commonsense reasoning benchmarks: ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), BoolQ (Clark et al., 2019), OpenbookQA (Mihaylov et al., 2018), RTE (Dagan et al., 2005), COPA (Roemmele et al., 2011), and RACE (Lai et al., 2017).

Evaluation framework. All perplexity and QA benchmarks are evaluated using the `lm-evaluation-harness` library from `https://github.com/EleutherAI/lm-evaluation-harness`, following the default evaluation protocols.

D. Extended results: calibration size

We study the effect of the calibration set size on the performance of `Ghosted Layers` by varying the number of sequences used to estimate $\mathbf{W}^*$. All sequences are sampled from C4 with length T=2,048, and we evaluate perplexity on WikiText-2, C4, and PTB under 7-layer pruning of LLaMA-3.1-8B with the LLM-Streamline criterion. As shown in Table A3, accuracy saturates almost immediately: 32 sequences already reach 60.01 AVG accuracy, and further increases to 64 or 128 yield essentially no gain (59.99 and 60.04).

Table A3. Ablation on the number of calibration sequences used to estimate $\mathbf{W}^*$, evaluated on LLaMA-3.1-8B with 7 out of 32 layers pruned under the LLM-Streamline criterion. Perplexity (↓) is reported on WikiText-2, C4, and PTB; PPL AVG is the mean across the three. Acc AVG (↑) denotes the mean zero-shot accuracy across the nine commonsense reasoning benchmarks used in our main experiments.

Num. of sequences	WikiText-2	C4	PTB	PPL AVG ↓	Acc AVG ↑
16	23.92	23.24	43.41	30.19	59.31
32	**21.35**	**21.52**	**40.56**	**27.81**	**60.01**
64	20.23	21.52	39.94	27.23	59.99
128	19.68	21.21	36.77	25.89	60.04

Perplexity continues to improve modestly with more calibration data, but the marginal returns diminish sharply beyond 32. We adopt 32 sequences as the default since this is the smallest size at which downstream accuracy is already saturated, and the additional perplexity reduction from larger calibration sets does not translate into accuracy gains.

E. Extended results: fine-tuning

Table A4. Comparison on Commonsense Reasoning benchmark with SOTA post-training method LLM-Streamline and Linear Patch 7-layer pruning

Model	Method	ARC-E	ARC-C	HellaS	WinoG	BoolQ	OBQA	RTE	CoPa	Race	AVG
LLaMA-2-7B	Dense	74.49	46.25	75.99	68.90	77.71	44.20	62.82	87.00	39.62	64.11
	Pruned LLM	55.89	36.18	62.64	66.38	62.17	37.20	52.35	81.00	33.78	54.18
	Linear Patch (Diag) + FT	61.62	37.63	68.61	68.19	72.02	36.60	68.59	84.00	37.51	59.42
	Linear Patch (Rotate) + FT	61.95	37.97	68.59	68.27	71.83	37.00	66.79	86.00	38.28	59.63
	Ghost Layer (Ours) + FT	62.92	36.69	67.87	67.01	75.66	37.40	67.15	85.00	37.70	59.71
LLaMA3-8B	Dense	77.65	53.41	79.16	72.38	81.35	45.00	69.68	89.00	40.00	67.51
	Pruned LLM	39.69	28.84	33.18	55.49	38.07	29.60	57.40	60.00	24.02	40.70
	Linear Patch (Diag) + FT	64.44	44.11	71.17	72.38	69.51	37.00	65.70	83.00	37.42	60.53
	Linear Patch (Rotate) + FT	64.60	44.28	71.09	73.16	70.12	37.00	66.06	83.00	37.22	60.73
	Ghost Layer (Ours) + FT	67.59	44.11	69.19	72.22	75.54	40.00	63.90	85.00	36.65	61.58
LLaMA3.1-8B	Dense	81.19	53.41	78.92	73.64	82.11	44.80	69.68	87.00	39.14	67.77
	Pruned LLM	44.19	33.11	33.39	56.99	38.20	32.60	58.12	61.00	25.84	42.60
	Linear Patch (Diag) + FT	67.42	43.26	70.93	72.14	67.80	36.60	69.31	81.00	38.37	60.76
	Linear Patch (Rotate) + FT	67.80	43.77	70.88	72.22	67.13	36.80	69.31	81.00	37.80	60.75
	Ghost Layer (Ours) + FT	70.03	43.69	68.81	71.67	73.09	40.20	67.87	84.00	36.75	61.79
D-LLaMA-8B	Dense	65.87	42.41	74.35	67.80	82.91	41.20	69.68	89.00	41.63	63.87
	Pruned LLM	49.92	35.24	46.33	61.56	51.99	32.40	57.40	70.00	27.27	48.01
	Linear Patch (Diag) + FT	56.48	37.46	67.92	64.80	81.28	37.80	70.04	83.00	38.85	59.74
	Linear Patch (Rotate) + FT	56.48	37.54	67.72	68.11	80.15	36.40	68.95	83.00	38.37	59.64
	Ghost Layer (Ours) + FT	57.37	38.91	66.01	66.77	77.83	37.80	72.92	87.00	39.33	60.44

E.1. Fine-tuning setup

We follow the fine-tuning protocol of `LinearPatch` (Chen et al., 2026) exactly to ensure a fair head-to-head comparison under identical post-training conditions. Specifically, we adopt a memory-efficient offline knowledge distillation strategy where the pruned model (student) is trained to match the output distribution of the unpruned model (teacher) via Kullback–Leibler (KL) divergence on the top-K logits.

Distillation objective. We optimize only the boundary operator ($\mathbf{W}^*$ for `Ghosted Layers` or $\mathbf{W}_{\mathrm{LP}}$ for `LinearPatch`) while freezing all other parameters of the pruned model. For each training sample $\mathbf{x} \in \mathcal{T}$, we minimize:

$$\min_{\mathbf{W}} \mathbb{E}_{\mathbf{x}\in\mathcal{T}}\, \mathrm{KL}(\mathbf{o}_t(\mathbf{x}),\ \mathbf{o}_s(\mathbf{x}))\,, \tag{A14}$$

where $\mathbf{o}_t$ and $\mathbf{o}_s$ denote the top-K logits probability distributions from the teacher and student, respectively, using the teacher's vocabulary indices for both. Following LinearPatch paper (Chen et al., 2026), we set $K = 100$.

Training configuration. We use the identical configuration across all compared methods (`LinearPatch (Diag) + FT`, `LinearPatch (Rotate) + FT`, and `Ghosted Layers + FT`) to isolate the contribution of the operator parameterization:

- **Optimizer:** AdamW (Loshchilov & Hutter, 2019) with a learning rate of 1×10^{-4}.
- **Training data:** 5,000 sequences of length $T = 2{,}048$ randomly sampled from the C4 training split (Raffel et al., 2020).
- **Schedule:** One epoch, with no learning rate warmup or decay.
- **Frozen parameters:** only the boundary operator is updated; all other model parameters are frozen.

Table A5. Comparison on PPL and Commonsense Reasoning benchmark with SOTA post-training method LLM-Streamline and Linear Patch 7-layer pruning

Model	Method	WIKI	C4	PTB	PPL AVG
LLaMA-2-7B	Dense	5.47	6.71	22.51	11.56
	Pruned LLM	18.45	25.37	62.18	35.33
	Linear Patch (Diag) + FT	11.13	12.19	37.09	20.14
	Linear Patch (Rotate) + FT	10.91	12.07	37.28	20.09
	Ghost Layer (Ours) + FT	10.01	10.31	31.27	17.20
LLaMA3-8B	Dense	6.14	8.61	10.58	8.44
	Pruned LLM	2305.48	2106.64	4642.40	3018.17
	Linear Patch (Diag)+ FT	17.29	21.59	30.25	23.04
	Linear Patch (Rotate)+ FT	17.15	21.62	30.48	23.08
	Ghost Layer (Ours) + FT	13.00	14.96	23.03	17.00
LLaMA3.1-8B	Dense	6.24	8.68	10.58	8.50
	Pruned LLM	2301.46	1173.53	3720.23	2398.41
	Linear Patch (Diag) + FT	17.26	21.51	30.20	22.99
	Linear Patch (Rotate) + FT	17.05	21.54	30.33	22.97
	Ghost Layer (Ours) + FT	13.10	14.99	23.14	17.08
D-LLaMA-8B	Dense	13.13	19.46	22.28	18.29
	Pruned LLM	3083.95	1291.64	2985.68	2453.76
	Linear Patch (Diag) + FT	35.96	34.63	51.01	40.53
	Linear Patch (Rotate) + FT	33.64	33.47	51.39	39.50
	Ghost Layer (Ours) + FT	22.92	25.26	33.66	27.28

For `Ghosted Layers`, the closed-form $\mathbf{W}^* = \mathbf{I} + \mathbf{M}^*$ is used as the initialization and then fine-tuned under the same objective. We drop any structural constraint on $\mathbf{W}^*$ during fine-tuning, consistent with the unconstrained formulation in Section 3.

Evaluation. We evaluate fine-tuned models on the same nine commonsense QA benchmarks (Table A4) and three perplexity corpora, WikiText-2 (Merity et al., 2017), C4 (Raffel et al., 2020), and PTB (Marcus et al., 1993) (Table A5), used throughout the main paper. Perplexity is computed on non-overlapping windows of length $T = 2{,}048$.

E.2. Fine-tuning Results

Tables A4 and A5 report the QA accuracy and perplexity comparisons between `Ghosted Layers + FT` and `LinearPatch + FT` across four LLM backbones (LLaMA-2-7B, LLaMA-3-8B, LLaMA-3.1-8B, and DeepSeek-R1-Distill-LLaMA-8B) with 7 layers pruned.

QA accuracy. As shown in Table A4, `Ghosted Layers + FT` attains the highest average accuracy across all four backbones under the same fine-tuning budget. The margin over the stronger `LinearPatch (Rotate) + FT` ranges from $+0.08$ on LLaMA-2-7B to $+1.04$ on LLaMA-3.1-8B, suggesting that the unconstrained operator continues to benefit from lightweight post-training, despite the fine-tuned `LinearPatch` variants being free to move out of the symmetric subspace during training.

Perplexity. Table A5 shows a more pronounced gap on the language modeling benchmarks. `Ghosted Layers + FT` achieves the lowest average perplexity across all four backbones, with relative reductions of 14.4% (LLaMA-2-7B), 26.3% (LLaMA-3-8B), 25.7% (LLaMA-3.1-8B), and 30.9% (DeepSeek-R1-Distill-LLaMA-8B) over the stronger `LinearPatch (Rotate) + FT`. This pattern is consistent with the interpretation that the closed-form $\mathbf{W}^*$ provides a lower-error initialization on the boundary alignment objective, and that starting distillation from this initialization yields better generative quality within the same training budget.

F. Extended results: Alternative calibration datasets

To evaluate the robustness of `Ghosted Layers` to the choice of calibration corpus, we repeat our main zero-shot QA experiments with the calibration dataset replaced from C4 (Raffel et al., 2020) to WikiText-2 (Merity et al., 2017). All other settings, including the number of calibration sequences (128), sequence length (T=2,048), pruning criterion (LLM-Streamline (Chen et al., 2025a)), and evaluation protocol across nine commonsense QA benchmarks, are identical to the main experiments reported in Table 1.

Table A6 reports the results across three LLM backbones and two pruning ratios. `Ghosted Layers` attains the highest average accuracy in all six settings, consistent with the C4-calibrated results in the main paper. The average accuracy of `Ghosted Layers` changes by at most 0.36 points when the calibration corpus is switched from C4 to WikiText-2 (e.g., $60.01 \rightarrow 59.65$ on LLaMA-3.1-8B at 7-layer pruning), indicating that the closed-form operator is not sensitive to the specific calibration distribution. Notably, the gap over the strongest training-free baseline widens under more aggressive 11-layer pruning, where the boundary activation mismatch is larger, suggesting that the unconstrained formulation is especially beneficial when the gap to reconstruct grows. This robustness to the calibration source is a practical advantage: practitioners can use whichever in-domain corpus is most readily available without retuning the recovery operator.

Table A6. Zero-shot accuracy (%) on nine commonsense QA benchmarks for 7-layer and 11-layer pruning across three LLM backbones, using **WikiText-2** as the calibration corpus instead of C4. All other settings match Table 1: 128 calibration sequences with sequence length T=2,048 and pruning boundaries selected via LLM-Streamline. AVG denotes the mean accuracy across all nine tasks. `Ghosted Layers` attains the highest average accuracy in all six settings, demonstrating robustness to the choice of calibration corpus.

Model	L_p/L_t	Method	ARC-E	ARC-C	HellaS	WinoG	BoolQ	OBQA	RTE	CoPa	Race	AVG ↑
LLaMA-3-8B	0/32	Dense	77.78	53.24	79.16	72.53	81.38	45.00	69.68	89.00	40.19	67.55
	7/32	Shortened LLaMA	58.88	32.68	59.17	54.06	45.44	34.40	54.15	75.00	30.72	49.39
	7/32	LLM-Streamline	39.65	29.18	33.23	55.41	38.04	29.80	57.40	60.00	24.02	40.75
	7/32	ShortGPT	39.65	29.18	33.23	55.41	38.04	29.80	57.40	60.00	24.02	40.75
	7/32	Prune&Comp	42.09	29.61	41.36	59.43	51.04	33.40	59.93	68.00	27.27	45.79
	7/32	ReplaceMe (LS)	64.44	43.69	64.32	72.45	67.65	37.00	68.95	75.00	35.41	58.77
	7/32	ReplaceMe (Cos)	50.93	34.73	49.71	66.14	39.02	34.00	63.18	66.00	29.67	48.15
	7/32	Linear Patch (D)	43.18	31.66	43.14	60.62	57.31	33.80	64.98	68.00	28.52	47.91
	7/32	Linear Patch (R)	51.14	34.13	49.24	63.14	57.25	34.00	67.51	67.00	29.76	50.35
	7/32	Ghost Layer (Ours)	65.70	43.86	66.33	72.30	69.60	38.40	67.87	78.00	36.94	59.89
	11/32	Shortened LLaMA	45.50	29.69	48.81	52.64	59.42	29.20	49.82	67.00	26.60	45.41
	11/32	LLM-Streamline	38.17	29.86	32.93	56.75	56.09	30.20	70.04	57.00	27.27	44.26
	11/32	ShortGPT	38.17	29.86	32.93	56.75	56.09	30.20	70.04	57.00	27.27	44.26
	11/32	Prune&Comp	37.25	28.16	42.86	57.14	62.81	29.20	56.32	62.00	25.07	44.53
	11/32	ReplaceMe (LS)	44.36	34.98	45.10	67.09	67.06	31.40	64.26	72.00	29.09	50.59
	11/32	ReplaceMe (Cos)	42.68	33.19	37.64	57.77	61.90	29.00	62.09	57.00	29.86	45.68
	11/32	Linear Patch (D)	46.51	35.58	47.68	60.77	70.89	32.00	69.31	69.00	30.24	51.33
	11/32	Linear Patch (R)	47.81	34.64	44.03	60.85	76.09	31.60	66.43	67.00	31.96	51.16
	11/32	Ghost Layer (Ours)	49.12	34.81	51.16	68.98	77.31	31.60	66.43	74.00	32.34	53.97
LLaMA-3.1-8B	0/32	Dense	81.31	53.50	78.90	73.72	82.02	44.80	69.68	87.00	39.23	67.80
	7/32	Shortened LLaMA	61.36	32.94	59.52	53.91	43.70	35.40	50.90	76.00	31.20	49.44
	7/32	LLM-Streamline	44.15	33.02	33.40	56.83	38.20	32.60	58.12	61.00	26.03	42.59
	7/32	ShortGPT	58.29	42.15	64.93	68.27	62.02	34.60	69.31	80.00	34.35	57.10
	7/32	Prune&Comp	45.20	30.46	44.05	58.41	51.41	34.80	60.65	67.00	27.46	46.60
	7/32	ReplaceMe (LS)	66.67	43.09	64.23	73.01	65.72	35.00	71.12	77.00	35.41	59.03
	7/32	ReplaceMe (Cos)	55.81	36.09	49.86	63.69	39.33	36.00	60.29	69.00	30.91	49.00
	7/32	Linear Patch (D)	48.06	32.76	46.46	61.40	60.34	35.20	68.23	68.00	29.00	49.94
	7/32	Linear Patch (R)	58.00	37.54	54.21	64.17	60.49	35.40	69.68	69.00	29.09	53.06
	7/32	Ghost Layer (Ours)	68.69	42.58	66.21	72.30	66.09	36.80	71.48	75.00	37.70	59.65
	11/32	Shortened LLaMA	48.53	29.61	49.52	52.72	59.36	29.60	51.26	64.00	26.51	45.68
	11/32	LLM-Streamline	39.65	30.29	31.47	56.51	55.08	30.20	69.68	61.00	28.52	44.71
	11/32	ShortGPT	39.65	30.29	31.47	56.51	55.08	30.20	69.68	61.00	28.52	44.71
	11/32	Prune&Comp	42.47	29.27	42.88	56.20	63.79	29.00	59.93	61.00	27.37	45.77
	11/32	ReplaceMe (LS)	46.00	35.32	44.48	66.85	65.66	32.00	70.40	75.00	29.57	51.70
	11/32	ReplaceMe (Cos)	43.69	32.17	36.44	57.30	58.47	29.40	68.95	62.00	31.48	46.66
	11/32	Linear Patch (D)	50.51	36.77	48.29	62.51	70.61	31.60	72.56	68.00	29.76	52.29
	11/32	Linear Patch (R)	50.80	35.15	46.60	61.88	75.29	31.00	70.04	70.00	30.72	52.39
	11/32	Ghost Layer (Ours)	50.46	34.30	50.62	68.35	74.65	32.40	67.87	72.00	33.11	53.75
DeepSeek-R1-Distill-LLaMA-8B	0/32	Dense	65.91	42.49	74.35	67.88	82.91	41.40	69.68	89.00	41.53	63.91
	7/32	Shortened LLaMA	48.48	30.97	50.84	50.75	62.72	30.60	58.48	69.00	31.67	48.17
	7/32	LLM-Streamline	49.12	37.12	55.98	63.22	77.06	34.00	74.73	71.00	33.21	55.05
	7/32	ShortGPT	49.12	37.12	55.98	63.22	77.06	34.00	74.73	71.00	33.21	55.05
	7/32	Prune&Comp	47.22	31.48	53.75	55.88	72.60	32.20	65.70	63.00	32.25	50.45
	7/32	ReplaceMe (LS)	54.34	37.37	60.04	66.30	73.61	33.40	72.56	81.00	40.10	57.64
	7/32	ReplaceMe (Cos)	51.47	36.09	58.12	62.67	76.27	31.00	75.45	71.00	34.16	55.14
	7/32	Linear Patch (D)	54.71	36.43	60.62	64.80	81.35	33.40	74.73	72.00	36.08	57.12
	7/32	Linear Patch (R)	53.87	36.52	60.77	66.14	79.66	34.60	74.73	74.00	37.51	57.53
	7/32	Ghost Layer (Ours)	54.12	37.12	59.54	67.32	70.95	33.40	76.17	81.00	40.04	57.74
	11/32	Shortened LLaMA	39.52	26.28	37.90	50.43	40.98	24.80	52.35	61.00	25.36	39.85
	11/32	LLM-Streamline	38.17	29.86	32.93	56.75	56.09	30.20	70.04	57.00	27.27	44.26
	11/32	ShortGPT	37.12	31.57	38.72	55.25	75.23	27.40	64.26	63.00	26.32	46.54
	11/32	Prune&Comp	35.98	29.35	36.54	51.93	64.53	25.60	59.57	56.00	24.69	42.69
	11/32	ReplaceMe (LS)	40.49	32.34	44.05	61.56	81.80	31.80	74.37	68.00	32.15	51.84
	11/32	ReplaceMe (Cos)	38.47	31.06	40.85	54.22	77.77	27.20	67.15	65.00	30.14	47.98
	11/32	Linear Patch (D)	45.12	34.22	43.12	57.85	77.40	32.00	67.87	61.00	28.61	49.69
	11/32	Linear Patch (R)	43.52	32.42	44.96	59.91	77.98	30.60	69.68	65.00	31.00	50.56
	11/32	Ghost Layer (Ours)	43.94	33.36	46.43	62.83	76.67	33.20	76.90	71.00	34.45	53.20

G. Extended results: Additional LLM architectures

To evaluate the generality of `Ghosted Layers` beyond the backbones reported in the main paper, we extend our zero-shot QA experiments to three additional LLMs: OLMo-2-7B (Walsh et al., 2025), LLaMA-2-7B (Touvron et al., 2023b), and Qwen-3-14B (Yang et al., 2025). These models span different pretraining corpora, model generations, and scales, allowing us to assess whether the recovery quality of `Ghosted Layers` transfers across architectural and training variations. All other experimental settings—including the calibration corpus (128 sequences from C4, T=2,048), pruning criterion (LLM-Streamline (Chen et al., 2025a)), and evaluation protocol across nine commonsense QA benchmarks—are identical to the main experiments in Table 1. For Qwen-3-14B, which has L=40 layers, we report results at $13/40$ and $15/40$ pruning ratios to match the relative pruning depths of $7/32$ and $11/32$ used for the 32-layer backbones.

Table A7 reports the results. `Ghosted Layers` attains the highest average accuracy in all six settings across the three backbones and two pruning ratios, consistent with the trend observed in the main paper. Notably, the margin over the strongest training-free baseline tends to widen on Qwen-3-14B under aggressive pruning ($15/40$), where `Ghosted Layers` achieves 43.61 AVG accuracy versus 40.08 for the next-best method (`ReplaceMe (Cos)`). The relative ordering among baselines is similar to that in Table 1, indicating that `Ghosted Layers` generalizes favorably across architectures with different pretraining corpora, scales, and design choices, including grouped-query attention variants such as Qwen-3.

Table A7. Zero-shot accuracy (%) on nine commonsense QA benchmarks for three additional LLM backbones beyond those in the main paper: OLMo-2-7B, LLaMA-2-7B, and Qwen-3-14B. All methods are training-free and use 128 calibration sequences sampled from C4 with sequence length T=2,048. Pruning boundaries are selected via the LLM-Streamline criterion. AVG denotes the mean accuracy across all nine tasks. `Ghosted Layers` achieves the highest average accuracy across all three backbones and both pruning ratios.

Model	L_p/L_t	Method	ARC-E	ARC-C	HellaS	WinoG	BoolQ	OBQA	RTE	CoPa	Race	AVG ↑
OLMo-2-7B	0/32	Dense	81.19	56.14	78.93	74.66	78.23	45.20	71.12	89.00	40.10	68.29
	7/32	Shortened LLaMA	71.34	43.26	68.16	65.75	65.20	41.20	62.09	79.00	33.30	58.81
	7/32	LLM-Streamline	66.41	40.78	64.20	70.32	69.63	36.00	71.48	82.00	37.70	59.84
	7/32	ShortGPT	60.65	37.37	62.81	67.09	38.13	37.00	54.51	73.00	33.30	51.54
	7/32	Prune&Comp	43.56	26.37	44.68	57.30	54.68	28.80	49.10	69.00	27.08	44.51
	7/32	ReplaceMe (LS)	65.70	39.59	63.14	70.96	70.21	36.40	70.04	81.00	38.09	59.46
	7/32	ReplaceMe (Cos)	66.88	40.10	63.65	70.40	67.74	36.40	73.29	81.00	38.09	59.73
	7/32	Linear Patch (D)	62.46	38.40	67.78	70.17	43.88	37.40	64.62	78.00	33.68	55.15
	7/32	Linear Patch (R)	67.34	41.64	67.01	71.03	63.94	37.20	67.87	80.00	36.27	59.14
	7/32	Ghost Layer (Ours)	67.67	41.81	63.62	70.88	70.24	37.87	70.04	80.00	37.42	59.95
	11/32	Shortened LLaMA	53.45	30.29	52.32	54.70	62.23	32.20	53.79	66.00	31.58	48.51
	11/32	LLM-Streamline	50.17	33.96	52.64	67.01	56.33	33.60	78.34	71.00	30.43	52.61
	11/32	ShortGPT	36.78	25.43	33.59	51.85	38.32	26.80	50.54	66.00	25.93	39.47
	11/32	Prune&Comp	28.79	25.68	27.36	50.75	39.27	27.00	47.65	60.00	20.86	36.37
	11/32	ReplaceMe (LS)	46.34	31.91	46.22	65.90	68.04	31.40	73.65	70.00	30.33	51.53
	11/32	ReplaceMe (Cos)	49.79	33.62	51.68	66.85	60.09	32.80	78.34	69.00	29.86	52.45
	11/32	Linear Patch (D)	39.86	30.55	42.43	56.99	51.19	34.80	71.84	64.00	26.99	46.52
	11/32	Linear Patch (R)	50.38	35.15	50.71	61.48	59.30	37.40	76.90	74.00	30.53	52.87
	11/32	Ghost Layer (Ours)	46.51	33.87	45.45	67.25	75.99	33.00	77.26	68.00	30.81	53.13
LLaMA-2-7B	0/32	Dense	74.49	46.25	75.99	68.90	77.71	44.20	62.82	87.00	39.62	64.11
	7/32	Shortened LLaMA	43.77	26.88	42.98	50.83	57.03	31.00	51.26	75.00	27.94	45.19
	7/32	LLM-Streamline	48.61	32.76	56.15	64.48	62.17	32.80	57.40	77.00	32.25	51.51
	7/32	ShortGPT	48.61	32.76	56.15	64.48	62.17	32.80	57.40	77.00	32.25	51.51
	7/32	Prune&Comp	48.44	31.91	53.81	59.83	62.17	35.80	57.04	83.00	31.96	51.55
	7/32	ReplaceMe (LS)	50.55	35.07	54.93	64.64	65.35	33.40	58.48	74.00	35.79	52.47
	7/32	ReplaceMe (Cos)	49.92	33.79	57.18	64.88	62.17	33.40	62.09	76.00	33.30	52.53
	7/32	Linear Patch (D)	55.13	34.56	57.12	63.46	62.17	35.60	55.96	78.00	35.02	53.00
	7/32	Linear Patch (R)	55.22	33.70	57.92	65.19	62.14	35.60	55.60	78.00	34.64	53.11
	7/32	Ghost Layer (Ours)	52.99	33.79	58.01	66.38	70.28	35.80	53.43	76.00	37.70	53.82
	11/32	Shortened LLaMA	44.95	25.09	42.44	51.07	47.77	30.40	54.87	72.00	27.08	43.96
	11/32	LLM-Streamline	42.59	32.59	48.43	62.35	62.23	30.40	58.48	78.00	30.33	49.49
	11/32	ShortGPT	42.80	30.46	44.50	60.46	62.26	35.40	47.29	72.00	29.57	47.19
	11/32	Prune&Comp	42.68	28.16	42.79	57.70	62.26	30.00	52.71	78.00	25.93	46.69
	11/32	ReplaceMe (LS)	40.99	32.17	46.29	60.54	74.59	29.80	56.68	73.00	32.34	49.60
	11/32	ReplaceMe (Cos)	38.80	32.94	46.57	57.62	62.14	30.40	60.29	74.00	31.58	48.26
	11/32	Linear Patch (D)	48.95	33.53	53.35	63.06	62.20	34.00	59.93	77.00	34.07	51.79
	11/32	Linear Patch (R)	49.28	32.51	52.73	62.98	62.17	33.20	61.73	77.00	33.88	51.72
	11/32	Ghost Layer (Ours)	49.81	31.83	49.28	65.82	73.70	33.80	57.04	72.00	33.01	51.81
Qwen-3-14B	0/40	Dense	82.83	60.24	78.82	72.85	89.30	46.20	77.62	90.00	43.16	71.22
	13/40	Shortened LLaMA	31.35	28.16	43.93	49.96	51.01	29.20	54.51	72.00	29.38	43.27
	13/40	LLM-Streamline	33.80	31.31	32.16	56.91	62.17	30.00	48.74	64.00	25.26	42.71
	13/40	ShortGPT	29.59	26.71	37.82	49.49	62.02	28.80	49.82	61.00	24.88	41.13
	13/40	Prune&Comp	33.80	31.31	32.16	56.91	62.17	30.00	48.74	64.00	25.26	42.71
	13/40	ReplaceMe (LS)	32.32	27.47	30.69	58.17	78.17	27.80	56.32	64.00	26.41	44.59
	13/40	ReplaceMe (Cos)	33.16	30.72	32.48	57.30	62.17	30.00	49.10	64.00	25.26	42.69
	13/40	Linear Patch (D)	25.08	22.70	25.04	49.57	37.83	27.60	52.71	55.00	25.93	35.72
	13/40	Linear Patch (R)	25.08	22.70	25.04	49.57	37.83	27.60	52.71	55.00	25.93	35.72
	13/40	Ghost Layer (Ours)	35.65	28.92	33.91	60.46	72.84	29.00	48.38	66.00	28.04	44.80
	15/40	Shortened LLaMA	35.27	23.04	34.16	50.36	52.87	27.40	55.23	59.00	26.70	40.45
	15/40	LLM-Streamline	39.48	27.39	35.85	48.22	38.50	29.00	50.18	61.00	21.05	38.96
	15/40	ShortGPT	29.04	26.54	33.49	48.70	61.68	28.60	52.35	55.00	23.83	39.91
	15/40	Prune&Comp	39.48	27.39	35.85	48.22	38.50	29.00	50.18	61.00	21.05	38.96
	15/40	ReplaceMe (LS)	38.59	25.68	34.89	51.46	37.80	29.40	46.57	60.00	26.99	39.04
	15/40	ReplaceMe (Cos)	42.72	27.82	38.41	47.51	38.26	30.00	50.18	62.00	23.83	40.08
	15/40	Linear Patch (D)	41.16	28.67	27.20	50.20	40.95	27.20	51.26	50.00	25.93	38.06
	15/40	Linear Patch (R)	37.58	26.96	35.69	50.67	43.58	29.60	49.46	55.00	25.93	39.39
	15/40	Ghost Layer (Ours)	44.78	27.39	43.57	53.91	43.39	30.60	51.99	67.00	29.86	43.61

H. Extended results: Effect of pruning depth on accuracy

Figure A1 shows the average commonsense accuracy on LLaMA-3.1-8B as the number of pruned layers increases from 7 to 15. Ghosted Layers consistently achieves the highest accuracy across all pruning depths. Notably, the performance gap over competing methods widens as pruning becomes more aggressive, indicating that the unconstrained operator remains effective even under larger activation mismatch.

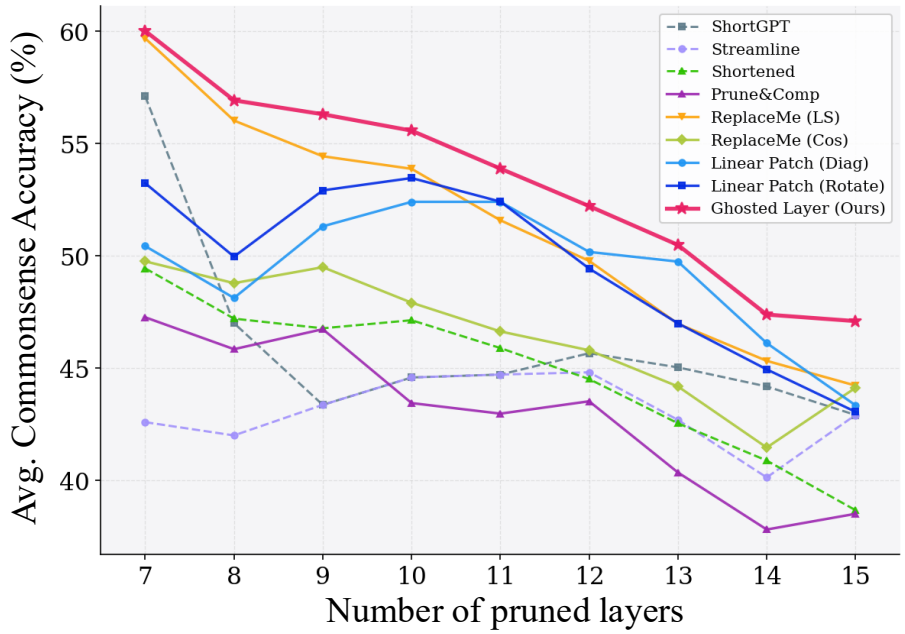

Figure A1. Average accuracy across 9 commonsense reasoning benchmarks with LLaMA-3.1-8B

I. Efficiency measurement

This section details the methodology and full experimental results for the inference efficiency measurements reported in Table 3.

I.1. Setup

Hardware. All measurements are conducted on a single NVIDIA A40 48GB GPU.

Model configuration. We evaluate LLaMA-3.1-8B under two pruning ratios, $n = 7$ and $n = 11$ out of $L = 32$ Transformer layers, in `float16`. Pruning boundaries are selected via LLM-Streamline (Chen et al., 2025a) using 128 calibration sequences of length $T = 2{,}048$ sampled from the C4 training split. The selected boundaries are $[23, 30)$ for $n = 7$ and $[19, 30)$ for $n = 11$.

Latency protocol. For each configuration, we measure prefill latency with 3 warmup iterations followed by 10 timed runs. Each run performs a single forward pass with `use_cache=False`, wrapped in `torch.cuda.synchronize()` before and after timing. We report the mean and standard deviation across the 10 runs.

GPU protocol. Peak GPU memory is measured via `torch.cuda.max_memory_allocated()`, which records the maximum activated tensor footprint during the forward pass and is unaffected by PyTorch's caching allocator reserving unused memory blocks. Before each measurement, we reset the peak counter via `torch.cuda.reset_peak_memory_stats()` and clear the allocator's cache through three cycles of `gc.collect()`, `torch.cuda.empty_cache()`, and `torch.cuda.ipc_collect()`, ensuring that the reported peak reflects only the memory required by the model under measurement.

Table A8. Prefill latency (ms) on LLaMA-3.1-8B with $n = 7$ pruned layers, across sequence lengths and batch sizes. Values are mean $\pm$ standard deviation over 10 runs with 3 warmup iterations; OOM indicates the configuration exceeded available GPU memory. `Ghosted Layers` consistently matches `LinearPatch` in latency across all configurations.

Seq	Batch	Dense	Pruned (Streamline)	LinearPatch (Diag)	LinearPatch (Rotate)	Ours
512	1	95.9 ± 0.2	76.6 ± 0.4	77.8 ± 0.4	78.8 ± 0.8	78.1 ± 0.2
512	4	352.3 ± 0.6	281.8 ± 1.6	284.2 ± 1.9	285.7 ± 1.6	285.5 ± 1.3
512	16	1329.0 ± 1.0	1055.4 ± 3.1	1064.6 ± 0.9	1067.8 ± 2.4	1071.0 ± 3.0
1024	1	185.7 ± 1.0	148.0 ± 2.2	148.1 ± 1.0	148.7 ± 1.0	149.4 ± 0.9
1024	4	704.3 ± 1.4	558.9 ± 1.1	564.1 ± 1.0	564.8 ± 1.1	564.8 ± 1.2
1024	16	2628.7 ± 3.3	2086.9 ± 2.3	2120.5 ± 3.6	2119.8 ± 1.1	2116.3 ± 2.4
2048	1	362.6 ± 1.6	287.8 ± 1.4	291.4 ± 1.6	291.7 ± 1.7	291.9 ± 1.7
2048	4	1359.8 ± 0.8	1079.8 ± 0.5	1093.9 ± 2.8	1090.5 ± 2.3	1091.5 ± 1.6
2048	16	5368.3 ± 4.6	4261.4 ± 3.5	4319.7 ± 3.4	4319.7 ± 4.8	4299.3 ± 7.6
4096	1	740.2 ± 1.7	587.8 ± 1.0	595.4 ± 0.8	594.5 ± 1.3	594.4 ± 1.0
4096	4	2778.7 ± 2.6	2206.2 ± 2.3	2230.5 ± 1.4	2232.0 ± 2.8	2231.9 ± 1.6
4096	16	11121.7 ± 5.1	8813.4 ± 13.2	8921.2 ± 9.7	8935.4 ± 5.9	8923.2 ± 2.7
8192	1	1506.0 ± 0.5	1193.2 ± 1.2	1205.3 ± 1.3	1206.2 ± 0.9	1205.5 ± 0.5
8192	4	5952.6 ± 4.2	4718.1 ± 2.0	4773.3 ± 5.3	4771.7 ± 3.8	4767.2 ± 3.6

I.2. Latency Analysis

Aggregate metrics. Table 3 in the main paper reports the aggregate GPU memory, prefill latency, accuracy, and perplexity at sequence length 2,048 and batch size 1. `LinearPatch` (both Diag and Rotate variants) and `Ghosted Layers` exhibit nearly identical wall-clock latency, reflecting the structural equivalence established in Section 4: both reduce to a single $C \times C$ matrix multiplication at the boundary. At $n = 7$, `Ghosted Layers` (291.9 ms) and `LinearPatch (Rotate)` (291.7 ms) are indistinguishable within measurement noise, and they remain within 0.5 ms of each other at $n = 11$.

Grid sweep across sequence lengths and batch sizes. To confirm that the cost equivalence holds beyond the representative configuration, we sweep prefill latency across sequence lengths $\{512, 1024, 2048, 4096, 8192\}$ and batch sizes $\{1, 4, 16\}$. Tables A8 and A9 report the $n = 7$ and $n = 11$ grids, respectively. Across both pruning ratios and all evaluated configurations, the latency of `Ghosted Layers` matches `LinearPatch` to within measurement noise and preserves

Table A9. Prefill latency (ms) on LLaMA-3.1-8B with $n = 11$ pruned layers, across sequence lengths and batch sizes. Same measurement protocol as Table A8.

Seq	Batch	Dense	Pruned (Streamline)	LinearPatch (Diag)	LinearPatch (Rotate)	Ours
512	1	96.5 ± 0.2	65.5 ± 0.2	73.6 ± 9.5	67.0 ± 0.2	66.4 ± 0.4
512	4	354.3 ± 2.1	238.6 ± 0.7	243.5 ± 0.3	243.3 ± 1.5	243.0 ± 1.6
512	16	1334.9 ± 2.0	894.2 ± 4.1	913.0 ± 2.9	910.1 ± 1.2	913.4 ± 3.1
1024	1	186.6 ± 1.7	124.8 ± 0.5	127.1 ± 1.1	127.0 ± 1.9	126.8 ± 0.9
1024	4	706.8 ± 1.7	476.2 ± 0.9	479.0 ± 1.4	479.8 ± 0.3	480.0 ± 0.4
1024	16	2636.6 ± 2.8	1785.9 ± 6.2	1804.5 ± 2.0	1808.8 ± 2.3	1804.1 ± 3.1
2048	1	363.5 ± 1.2	244.4 ± 1.4	248.1 ± 1.1	248.1 ± 1.0	247.6 ± 1.1
2048	4	1359.3 ± 0.9	917.4 ± 2.0	929.0 ± 1.3	929.9 ± 1.0	929.6 ± 0.5
2048	16	5349.4 ± 4.3	3607.1 ± 4.2	3675.6 ± 7.1	3681.4 ± 4.6	3674.7 ± 8.1
4096	1	739.8 ± 1.1	497.9 ± 1.0	505.0 ± 0.8	505.6 ± 1.0	505.8 ± 0.8
4096	4	2778.2 ± 1.9	1867.5 ± 2.9	1900.7 ± 1.2	1903.0 ± 2.6	1901.6 ± 2.0
4096	16	11115.8 ± 8.0	7479.6 ± 15.8	7600.3 ± 8.1	7618.9 ± 5.8	7608.7 ± 5.2
8192	1	1507.1 ± 0.4	1015.3 ± 0.7	1026.4 ± 0.8	1027.7 ± 1.2	1027.1 ± 0.7
8192	4	5945.9 ± 2.1	4017.7 ± 3.1	4059.2 ± 1.8	4066.5 ± 4.5	4063.6 ± 4.9

the speedup gained from layer removal regardless of sequence length or batch size. This confirms that the inference-cost equivalence established analytically in Section 4 holds robustly in practice.

J. Closed-form Solution Computation

The closed-form operator $\mathbf{M}^* = \mathbf{X}_{\text{pre}}^{\dagger}\mathbf{\Delta}$ in Theorem 4.1 can be obtained either by directly inverting the thin SVD of $\mathbf{X}_{\text{pre}}$ via `torch.linalg.svd`, or by solving the regularized normal equations via `torch.linalg.solve`. The latter yields a Tikhonov-regularized least-squares solution that converges to the unregularized pseudoinverse as $\epsilon \to 0$ (Golub & Van Loan, 2013); with the small $\epsilon = 10^{-6}$ used throughout, the two procedures coincide numerically when $\mathbf{X}_{\text{pre}}$ has full column rank, which holds with high probability under $T_{\mathcal{D}} \gg C$. Table A10 empirically confirms this numerical equivalence.

SVD-based. Given the thin SVD $\mathbf{X}_{\text{pre}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ with $\mathbf{U} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$, $\mathbf{\Sigma} \in \mathbb{R}^{C \times C}$, $\mathbf{V} \in \mathbb{R}^{C \times C}$ (the reduced form returned by `torch.linalg.svd` with `full_matrices=False`), the Moore–Penrose pseudoinverse is $\mathbf{X}_{\text{pre}}^{\dagger} = \mathbf{V}\mathbf{\Sigma}^{+}\mathbf{U}^{\top}$, yielding

$$\mathbf{M}^* = \mathbf{V}\mathbf{\Sigma}^{+}\mathbf{U}^{\top}\mathbf{\Delta}, \tag{A15}$$

where $\mathbf{\Sigma}^{+}$ truncates singular values below $10^{-6} \cdot \sigma_{\max}$ to zero for numerical stability (Higham, 2002). Internally, `torch.linalg.svd` computes the decomposition through a Jacobi-based driver (`gesvdj`) with a QR-based fallback (`gesvd`) on CUDA. This procedure is robust to rank deficiency, but explicitly materializes $\mathbf{U} \in \mathbb{R}^{T_{\mathcal{D}} \times C}$, whose size scales linearly with the calibration length $T_{\mathcal{D}}$.

Solver-based. The same minimum-norm solution can be obtained by solving the regularized normal equations

$$(\mathbf{X}_{\text{pre}}^{\top}\mathbf{X}_{\text{pre}} + \epsilon\mathbf{I})\,\mathbf{M}^* = \mathbf{X}_{\text{pre}}^{\top}\mathbf{\Delta}, \qquad \epsilon = 10^{-6}, \tag{A16}$$

via `torch.linalg.solve`, which solves the square linear system $\mathbf{A}\mathbf{X} = \mathbf{B}$ for invertible $\mathbf{A}$ (Golub & Van Loan, 2013). The small ridge term $\epsilon\mathbf{I}$ ensures the system remains well-conditioned and invertible, which is standard practice for numerically stable least-squares solves (Higham, 2002). Both accumulators $\mathbf{X}_{\text{pre}}^{\top}\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{pre}}^{\top}\mathbf{\Delta}$ live in $\mathbb{R}^{C \times C}$, with size independent of $T_{\mathcal{D}}$. They can therefore be accumulated in a streaming fashion across calibration batches, so the entire calibration corpus need never reside in memory at once.

Table A10. SVD-based and solver-based computation of $\mathbf{M}^*$ on LLaMA-3.1-8B with $n = 7$ pruned layers, using 32 calibration batches ($T_{\mathcal{D}} = 65{,}536$, $C = 4{,}096$) in `float64`. Both procedures produce identical results. ACC AVG (↑) denotes the mean zero-shot accuracy across the nine commonsense reasoning benchmarks used in our main experiments.

Method	WIKI	C4	PTB	PPL AVG	Acc AVG
torch.linalg.svd	21.35	21.53	40.56	27.81	60.00
torch.linalg.solve	21.35	21.52	40.56	27.81	60.01

As shown in Table A10, the two procedures yield negligible differences in both perplexity and accuracy across all benchmarks, confirming that the small ridge regularization $\epsilon = 10^{-6}$ has no measurable effect on downstream performance.