SHIFT-LLM
DISTRIBUTION SHIFT CORRECTION IN DEPTH-PRUNED LLM

Anonymous authors
Paper under double-blind review

ABSTRACT

Depth pruning removes entire Transformer blocks to reduce the inference cost of large language models, but disrupts the hidden-state distributions expected by downstream layers, leading to significant accuracy loss. We introduce SHIFT-LLM, a training-free post-pruning correction framework that inserts a Linear Residual Adapter (LRA) at each pruning site. Each LRA preserves the identity pathway of the original residual block and adds a lightweight affine residual correction. This correction is calibrated via closed-form least-squares regression on a small held-out set, without gradient computation, to approximate the missing residual update produced by the pruned block. Together with the preserved identity pathway, the resulting LRA output approximates the hidden state produced by the original block, thereby mitigating the distributional mismatch introduced by layer removal while avoiding the expensive attention and feed-forward computations of the removed blocks. The resulting LRAs support low-rank factorization and exact merging across consecutive pruned layers for additional compression, and combine naturally with parameter-efficient fine-tuning for further recovery beyond fine-tuning the pruned model alone. Experiments on five model families, six layer-selection criteria, and seven zero-shot benchmarks show that SHIFT-LLM consistently recovers accuracy lost to depth pruning across most configurations, achieving gains up to +15.7 points on Llama-3.1-8B-Instruct while requiring only a few hundred calibration samples and no gradient computation.

1 INTRODUCTION

Transformer-based Large Language Models (LLMs) (Vaswani et al., 2017) have achieved remarkable success across many applications, but their growing scale imposes substantial computational costs that hinder practical deployment in resource-constrained settings. To address this challenge, model compression methods such as quantization (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023), knowledge distillation (Gu et al., 2023; Xu et al., 2024), and pruning (Sun et al., 2023; Ashkboos et al., 2024) have been widely studied. Among them, structured pruning is especially attractive because it removes entire components, such as neurons or layers, enabling more direct hardware efficiency and inference speedup.

Pruning improves model efficiency by removing parameters or structural components while preserving performance as much as possible. Structured pruning (Chen et al., 2024; Liu et al., 2025; Cao et al., 2025; Ling et al.) removes entire neurons or layers rather than sparsifying individual weights (Ma et al., 2023; Sun et al., 2023), enabling more direct practical speedups and hardware efficiency. Recent studies further show that depth pruning can provide reliable small-batch latency gains while remaining competitive with structured width pruning (Kim et al., 2024; Men et al., 2025).

However, despite their practical appeal, existing depth pruning methods often suffer from significant performance degradation and require extensive fine-tuning to recover performance. In this work, we identify *post-pruning distribution shift* as a key underlying cause of this degradation. Specifically, removing an intermediate Transformer block eliminates the residual update that contributes to its output hidden state, causing downstream layers to receive hidden states that differ from those produced by the original pretrained model and therefore follow a shifted distribution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This shift can propagate through subsequent layers and lead to substantial performance degradation. Therefore, the challenge of depth pruning is not only to determine which blocks are removable, but also to mitigate the distribution shift induced by their removal.

This perspective also helps explain why post-pruning fine-tuning is effective. By adapting the remaining parameters to the shifted hidden-state distributions, fine-tuning can provide strong recovery, but requires additional optimization, compute, and training time. This motivates a lightweight alternative for settings where such post-pruning training is undesirable: directly approximating the missing residual update without gradient-based optimization. Moreover, when fine-tuning is available, this explicit correction can be used jointly with it to provide complementary recovery.

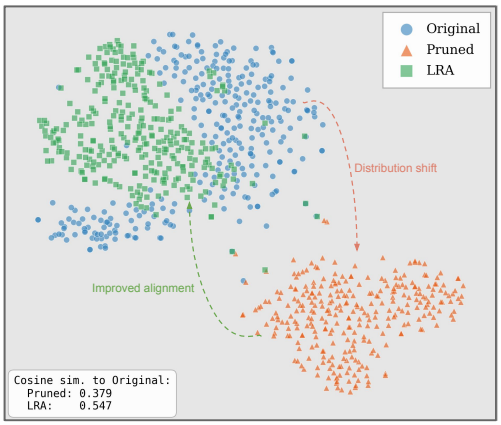


Figure 1: The proposed LRA partially restores the hidden-state distribution toward that of the original model.

Motivated by this observation, we introduce SHIFT-LLM, a *training-free* post-pruning correction framework for depth-pruned LLMs. Its core component is the Linear Residual Adapter (LRA), which preserves the identity pathway of the original residual block and adds a lightweight affine residual correction. This correction is calibrated via closed-form least-squares regression on a small held-out set, without gradient computation, to approximate the missing residual update. Together with the preserved identity pathway, the LRA output approximates the hidden state produced by the original block, thereby mitigating the post-pruning distribution shift while avoiding its expensive attention and feed-forward computations.

This formulation relies on two key design principles. First, existing layer-selection methods are designed to prune redundant or less critical layers, making their missing residual updates more amenable to low-complexity approximation and allowing a lightweight affine correction to capture their dominant effect. Second, the LRA preserves the residual structure of the original Transformer block by retaining the identity pathway and approximating only the missing residual update. This makes the approximation problem easier than directly reconstructing the full output hidden state, since the affine residual correction only needs to approximate the missing residual update. Moreover, retaining the identity pathway preserves the residual computation structure on which the original model was pretrained, allowing the corrected hidden state to better match the output of the original block. Figure 1 supports this view: the pruned model exhibits a clear shift in hidden-state space, whereas the proposed LRA moves the hidden states substantially closer to the original distribution. For additional efficiency, the resulting LRAs can be compressed via low-rank factorization and merged across consecutive pruning sites. Overall, these results suggest that approximating the missing residual updates while preserving the residual structure provides a simple, general, and effective complement to depth pruning. Our contributions are summarized as follows:

- We introduce SHIFT-LLM, a *training-free* framework that explicitly corrects the distribution shift caused by structured depth pruning in LLMs. Its core Linear Residual Adapter (LRA) preserves the identity pathway of the original residual block, while its affine residual correction approximates the missing residual update introduced by layer removal.

- SHIFT-LLM is a general and modular post-pruning correction framework that can be applied across a wide range of existing layer-selection strategies for depth pruning, without being tied to any specific pruning criterion.

- We conduct extensive experiments showing that SHIFT-LLM provides strong training-free recovery across diverse LLMs, pruning strategies, and evaluation settings, while remaining complementary to post-pruning fine-tuning and further improving its performance in most evaluated settings.

2 Related Works

LLM layer pruning aims to improve inference efficiency by removing selected Transformer blocks while preserving downstream performance. Existing methods typically rank layers using importance criteria based on weight statistics, gradient-based saliency, activation patterns, or the loss change caused by layer removal, and then prune the least important blocks to reduce computation and memory usage.

Several methods have been proposed for structured LLM pruning. LLM-Pruner (Ma et al., 2023) uses gradient-based importance with lightweight tuning, while LaCo (Yang et al., 2024) merges later layers into earlier ones. SlimGPT (Ling et al., 2024) uses non-uniform pruning to reduce cross-layer errors. Shortened LLaMA (Kim et al., 2024) and ShortGPT (Men et al., 2025) rank redundant layers using importance criteria and Block Influence, respectively. SlimLLM (Guo et al., 2025) further performs sub-module pruning with linear-regression-based recovery.

LLM-Streamline (Chen et al., 2024) prunes consecutive low-importance layers based on their effect on hidden states and uses a lightweight replacement module, together with a stability metric, to reduce performance loss. GRASP (Liu et al., 2025) preserves gradient-sensitive singular components and replaces redundant layers with a compact parameterization estimated from a small calibration set. ReplaceMe (Shopkhoev et al., 2025) is a concurrent training-free depth pruning method that compensates for removed Transformer blocks using a linear transformation fitted on a small calibration set. The key difference lies in the correction formulation: ReplaceMe regresses through a neighboring layer to approximate the effect of the removed block, whereas SHIFT-LLM performs a local correction at each pruning site by estimating only the missing residual update and preserving the original identity pathway. A detailed theoretical and empirical comparison is provided in the supplementary material. PIP (Cao et al., 2025) performs structured pruning by comparing clean and perturbed input views and using gradient differences to identify less sensitive components. Navigation LLM Layer Pruning (Lu et al., 2024) provides a large-scale empirical study showing that simple layer-pruning strategies combined with fine-tuning of the output head and recent layers can be highly effective. Finally, Siddiqui et al. (Siddiqui et al., 2024) study simple additive corrections and trained low-rank linear adapters in place of removed blocks, showing that lightweight linear modules can partially recover depth-pruning performance, albeit through gradient-based fitting rather than a closed-form training-free formulation.

3 Method

In this section, we introduce SHIFT-LLM, our training-free post-pruning correction framework for mitigating the hidden-state distribution shift caused by depth pruning. We first formalize the effect of layer removal on downstream hidden states, then present its core component, the Linear Residual Adapter (LRA). The LRA preserves the identity pathway of the original residual block and adds an affine residual correction that approximates the missing residual update. Together, they produce a corrected hidden state that approximates the output hidden state of the original block. We further derive the closed-form estimation procedure and discuss practical extensions, including low-rank compression and exact merging across consecutive pruning sites. Figure 2 provides an overview of SHIFT-LLM.

3.1 Pruning-Induced Hidden-State Shift

We consider a Transformer composed of residual blocks, where the hidden representation at layer ℓ is given by

$$h^{(\ell)} = h^{(\ell-1)} + f_\ell\Big(h^{(\ell-1)}\Big), \tag{1}$$

where $h^{(\ell-1)} \in \mathbb{R}^{T\times d}$ denotes the input hidden states, T is the sequence length, d is the hidden dimension, and $f_\ell(\cdot) : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ denotes the transformation implemented by the ℓ-th block, including the attention and feed-forward computations. In standard depth pruning, removing layer ℓ bypasses this transformation entirely. The resulting hidden states after pruning become

$$\tilde{h}^{(\ell)} = h^{(\ell-1)}, \tag{2}$$

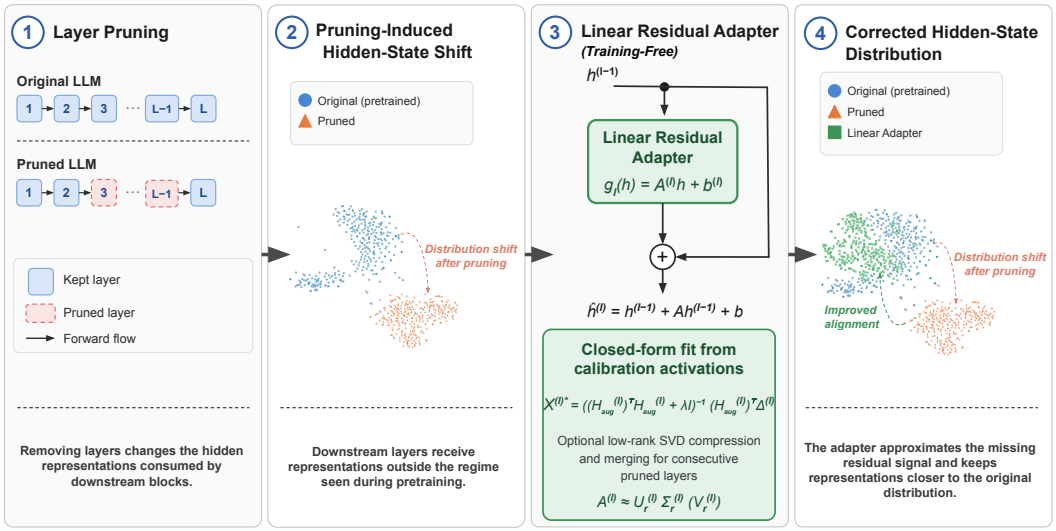


Figure 2: Overview of SHIFT-LLM for depth pruning in LLMs. Depth pruning induces a hidden-state distribution shift. At each pruning site, the LRA preserves the identity pathway and adds an affine residual correction estimated in closed form from calibration activations to approximate the missing residual update. The resulting corrected hidden states remain closer to the original distribution, while the LRAs support low-rank compression and exact merging across consecutive pruning sites.

instead of the original output $h^{(\ell)}$. Consequently, the subsequent block receives representations that differ systematically from those observed during pretraining. The missing residual update introduced by pruning can be written as

$$\delta^{(\ell)} = h^{(\ell)} - \tilde{h}^{(\ell)} = f_\ell\left(h^{(\ell-1)}\right), \qquad \delta^{(\ell)} \in \mathbb{R}^{T\times d}. \tag{3}$$

Equation equation 3 shows that depth pruning removes the residual update produced by the pruned block. As a result, downstream layers receive hidden states whose distribution differs from that produced by the original pretrained model. This shift is particularly harmful in deep residual architectures because the output hidden state of each block becomes the input to subsequent blocks. When layer ℓ is removed, block $\ell+1$ receives $\tilde{h}^{(\ell)}$ instead of the original hidden state $h^{(\ell)}$, and the resulting shift can propagate through the remaining network.

SHIFT-LLM is motivated by this observation. Rather than viewing layer pruning solely as the removal of redundant computation, we also consider the hidden-state distribution shift induced by the missing residual update. This suggests that an effective post-pruning correction should approximate the missing residual update so that the corrected hidden state remains close to that produced by the original block, while preserving the computational benefits of depth pruning.

3.2 Linear Residual Adapter Formulation

Motivated by the analysis above, we insert a lightweight LRA at each pruning site to mitigate the hidden-state distribution shift caused by layer removal. Suppose that layer ℓ is pruned from the original network. Rather than directly forwarding $h^{(\ell-1)}$ to the subsequent block, the LRA preserves this identity pathway and adds an affine residual correction $g_\ell(\cdot)$ that approximates the missing residual update. The corrected hidden state is therefore defined as

$$\hat{h}^{(\ell)} = h^{(\ell-1)} + g_\ell\left(h^{(\ell-1)}\right), \tag{4}$$

where $g_\ell(\cdot) : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ denotes the affine residual correction used to approximate the missing residual update $\delta^{(\ell)}$. Our objective is therefore to approximate the missing residual update $\delta^{(\ell)}$ defined in Eq. equation 3. To this end, we model the affine residual correction as

$$g_\ell(h) = hA^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top, \tag{5}$$

where $A^{(\ell)} \in \mathbb{R}^{d\times d}$, $b^{(\ell)} \in \mathbb{R}^d$, and $\mathbf{1} \in \mathbb{R}^T$ broadcasts the bias over all token positions. Substituting Eq. (5) into Eq. (4) yields

$$\hat{h}^{(\ell)} = h^{(\ell-1)} + h^{(\ell-1)}A^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top = h^{(\ell-1)}\left(I + A^{(\ell)}\right) + \mathbf{1}(b^{(\ell)})^\top. \tag{6}$$

The affine residual correction $g_\ell(\cdot)$ provides a local approximation of the missing residual update in Eq. equation 3. Importantly, the LRA does not reconstruct the entire output hidden state from scratch: it preserves the identity pathway and estimates only the contribution removed by pruning. This simplifies the approximation problem while retaining the residual computation structure on which the original model was pretrained. Together, the identity pathway and affine residual correction produce a corrected hidden state that approximates the output of the original block, thereby mitigating the pruning-induced distribution shift.

The LRA is designed to satisfy three practical requirements. First, it should be substantially cheaper than the original Transformer block so that the efficiency benefits of depth pruning are preserved. Second, its affine residual correction should be estimable from a small calibration set without gradient-based optimization. Third, it should operate locally at each pruning site without modifying the remaining network.

With this formulation, the pruned model no longer propagates the uncorrected hidden state $\tilde{h}^{(\ell)} = h^{(\ell-1)}$, but instead uses the corrected hidden state in Eq. equation 6. Since each LRA defines an affine mapping through the preserved identity pathway and affine residual correction, it naturally supports low-rank factorization and exact merging across consecutive pruning sites, as discussed in Sec. 3.4.

3.3 Closed-Form Estimation

The affine residual correction of the LRA is estimated from a small calibration set without gradient-based optimization. The original unpruned model is run once on the calibration samples, and for each pruned layer ℓ, the input hidden states and corresponding residual updates are collected via forward hooks. Stacking the token-level inputs yields $H^{(\ell)} \in \mathbb{R}^{N\times d}$, where each row corresponds to a sampled token representation from $h^{(\ell-1)}$, and N denotes the total number of collected tokens across all calibration samples. For the same samples, we stack the missing residual updates defined in Eq. equation 3 into $\Delta^{(\ell)} \in \mathbb{R}^{N\times d}$, with $\Delta^{(\ell)}_{n,:} = h^{(\ell)}_{n,:} - h^{(\ell-1)}_{n,:}$.

Using the affine residual correction defined in Eq. equation 5, our objective is to estimate $A^{(\ell)}$ and $b^{(\ell)}$ such that $g_\ell(h)$ approximates the missing residual update. To jointly estimate the weight matrix and bias term, we augment the input matrix with a column of ones:

$$H^{(\ell)}_{\text{aug}} = \left[H^{(\ell)} \mid \mathbf{1}\right] \in \mathbb{R}^{N\times(d+1)}, \tag{7}$$

and define the stacked parameter matrix

$$X^{(\ell)} = \begin{bmatrix} A^{(\ell)} \\ (b^{(\ell)})^\top \end{bmatrix} \in \mathbb{R}^{(d+1)\times d}. \tag{8}$$

The parameters of the affine residual correction are then obtained by solving the ridge-regression problem

$$X^{(\ell)*} = \arg\min_X \left\| H^{(\ell)}_{\text{aug}} X - \Delta^{(\ell)} \right\|_F^2 + \lambda \|X\|_F^2, \tag{9}$$

where $\lambda > 0$ is a regularization coefficient. The minimizer of Eq. (9) has the closed-form solution

$$X^{(\ell)*} = \left((H^{(\ell)}_{\text{aug}})^\top H^{(\ell)}_{\text{aug}} + \lambda I\right)^{-1} (H^{(\ell)}_{\text{aug}})^\top \Delta^{(\ell)}. \tag{10}$$

The first d rows of $X^{(\ell)*}$ define $A^{(\ell)}$, and the last row defines $(b^{(\ell)})^\top$. Once estimated, the LRA is applied as

$$\hat{h}^{(\ell)} = h^{(\ell-1)} + h^{(\ell-1)}A^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top. \tag{11}$$

This estimation procedure is entirely training-free: it requires only forward passes through the original model to collect calibration activations, followed by a single linear solve for each pruned layer. It therefore avoids iterative backpropagation, optimizer tuning, and recovery fine-tuning, while providing a closed-form estimate of the affine residual correction that approximates the missing residual update.

Table 1: Zero-shot average accuracy across seven evaluation benchmarks for different pruning criteria without fine-tuning at a pruning ratio of 25%. "Original" denotes the unpruned model before layer pruning. "Base" denotes the pruned model without post-pruning correction, while "+LRA" denotes the same pruned model equipped with the proposed LRA. "Gain" reports the average improvement obtained by adding the LRA. Full per-benchmark results for each model are provided in the supplementary material.

	Qwen2-1.5B			Qwen1.5-7B			Llama-3.1-8B-It			Vicuna-7B		
Original	59.87			65.54			71.03			66.80		
Pruning Criteria	Base	+LRA	Gain	Base	+LRA	Gain	Base	+LRA	Gain	Base	+LRA	Gain
Block Influence	45.78	**46.85**	+1.08	52.06	**52.82**	+0.76	57.18	**61.62**	+4.44	55.05	**55.52**	+0.48
Reverse-order*	43.71	**45.71**	+2.00	49.25	**53.79**	+4.54	43.33	**59.06**	+15.74	**53.47**	52.96	-0.51
Magnitude-ℓ_1	47.44	**48.47**	+1.03	40.72	**47.59**	+6.88	37.81	**39.42**	+1.61	34.86	**35.09**	+0.22
Magnitude-ℓ_2	43.76	**45.78**	+2.02	43.83	**45.74**	+1.91	36.00	**39.58**	+3.58	35.10	**37.03**	+1.92
Taylor	45.93	**46.91**	+0.98	51.19	**53.04**	+1.85	43.30	**59.02**	+15.71	56.66	**56.67**	+0.01
PPL	45.73	**47.76**	+2.03	35.89	**37.53**	+1.64	61.17	**62.21**	+1.05	55.34	50.92	-4.43

3.4 Extensions and Complexity

The affine structure of the proposed LRA enables two practical efficiency extensions without changing the closed-form estimation procedure. First, the fitted weight matrix $A^{(\ell)}$ can be compressed by truncated singular value decomposition,

$$A^{(\ell)} \approx U_r^{(\ell)} \Sigma_r^{(\ell)} (V_r^{(\ell)})^\top, \tag{12}$$

where $r \ll d$ is the retained rank. The affine residual correction can then be written as

$$g_\ell(h) \approx h P^{(\ell)} Q^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top, \tag{13}$$

with $P^{(\ell)} = U_r^{(\ell)} \Sigma_r^{(\ell)}$ and $Q^{(\ell)} = (V_r^{(\ell)})^\top$. This reduces the parameter count from $d^2 + d$ to $2dr + d$, which is particularly useful when pruned layers are non-consecutive and require separate local LRAs.

Second, consecutive LRAs can be merged exactly. Since each LRA defines an affine mapping, it can be written as

$$\hat{h}^{(\ell)} = h^{(\ell-1)} M^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top, \qquad M^{(\ell)} = I + A^{(\ell)}. \tag{14}$$

Because the composition of affine mappings remains affine, consecutive LRAs can be collapsed into a single transformation without introducing additional approximation. Thus, SHIFT-LLM supports both low-rank compression and exact merging while preserving the efficiency benefits of depth pruning, as further quantified through computational cost and runtime analysis in Sec. 4.4.

4 Experiments

In this section, we evaluate SHIFT-LLM across different model families, layer-selection criteria, and recovery settings. We first assess its training-free effectiveness without post-pruning fine-tuning, including zero-shot accuracy, language-modeling perplexity, and scaling to a 14B model. We then evaluate its compatibility with post-pruning fine-tuning and compare it with existing pruning pipelines. Finally, we analyze the main design choices and practical efficiency of the proposed framework. Additional experiments and implementation details are provided in the supplementary material.

4.1 Main Results Without Fine-Tuning

Table 1 summarizes the zero-shot performance of depth-pruned models without post-pruning fine-tuning across four LLMs, averaged over seven benchmarks. Overall, adding the proposed LRA improves the pruned baselines across most pruning criteria and models. The affine residual correction approximates the missing residual update, while the preserved identity pathway helps keep the corrected hidden state closer to that produced by the original block.

The LRA improves all pruning criteria on Qwen2-1.5B and Qwen1.5-7B, with gains up to **+2.03** and **+6.88**, respectively, and achieves gains of **+15.74** and **+15.71** on Llama-3.1-8B-Instruct under Reverse-order* and Taylor pruning. Gains are smaller and occasionally negative on Vicuna-7B, suggesting

Table 2: Zero-shot results with fine-tuning on C4 under Reverse-order* pruning. "Orig." denotes the unpruned model, "LoRA" denotes standard post-pruning LoRA fine-tuning, and "LoRA+LRA" denotes LoRA fine-tuning combined with the proposed LRA, whose affine residual correction is kept trainable during fine-tuning.

Dataset	Qwen2-1.5B			Qwen1.5-7B			Llama-3.1-8B-It			Vicuna-7B		
	Orig.	LoRA	LoRA+LRA	Orig.	LoRA	LoRA+LRA	Orig.	LoRA	LoRA+LRA	Orig.	LoRA	LoRA+LRA
BoolQ	73.36	63.15	**65.54**	81.9	71.28	**73.24**	83.98	**70.31**	69.11	80.55	74.34	**75.99**
PIQA	75.46	**64.31**	64.25	78.40	68.72	**69.91**	79.98	71.71	**72.25**	77.26	70.51	**71.16**
HellaSwag	65.41	45.64	**49.18**	76.97	61.70	**63.54**	79.27	67.95	**70.43**	73.76	66.14	**67.28**
WinoGrande	66.22	**58.41**	57.93	66.14	**59.98**	58.09	74.19	65.98	**66.14**	69.61	**65.19**	63.38
ARC-e	66.16	45.03	**46.30**	70.71	**58.75**	58.00	81.78	63.22	**66.37**	75.63	62.75	**63.80**
ARC-c	36.09	28.16	**30.38**	42.83	35.32	**38.57**	55.03	43.52	**44.71**	45.82	39.76	**40.27**
OBQA	36.40	28.00	**29.00**	41.80	**32.80**	32.60	43.00	37.20	**38.80**	45.00	37.00	**38.80**
Avg.	59.87	47.53	**48.94**	65.54	55.51	**56.28**	71.03	59.98	**61.12**	66.80	59.38	**60.10**

that some pruning criteria may select blocks whose missing residual updates remain too complex for the token-wise affine correction. Supplementary experiments support this explanation: changing the calibration domain does not remove these negative gains, while random layer selection further degrades performance, indicating that SHIFT-LLM is most effective when the pruning criterion identifies blocks whose residual updates are sufficiently amenable to low-complexity approximation. Overall, the results show that SHIFT-LLM provides effective training-free correction across diverse pruning criteria and model families.

Scaling to Qwen2.5-14B. We further evaluate SHIFT-LLM on Qwen2.5-14B at 25% depth pruning. The LRA improves average zero-shot accuracy from 53.34 to 56.32 under Block Influence (**+2.98**) and from 46.72 to 52.08 under Reverse-order* pruning (**+5.36**). These results show that the proposed training-free correction remains effective at the 14B scale. Full per-benchmark results are provided in the supplementary material.

Language Modeling Evaluation. To evaluate performance beyond zero-shot classification, we measure WikiText-2 perplexity at 25% depth pruning while using C4 only for calibration. As shown in Table 3, the proposed LRA substantially reduces perplexity across both pruning criteria for Qwen2-1.5B, Qwen1.5-7B, and Llama-3.1-8B, and under BI pruning for Vicuna-7B. The exception is Vicuna-7B under Reverse-order* pruning, consistent with the weaker recovery observed in Table 1.

Model	Criterion	Base	+LRA	Gain
Qwen2-1.5B	BI	87.16	**52.98**	+34.18
	Reverse*	458.44	**111.33**	+347.11
Qwen1.5-7B	BI	146.65	**54.66**	+91.99
	Reverse*	130.21	**64.12**	+66.09
Llama-3.1-8B	BI	161.51	**33.83**	+127.68
	Reverse*	1179.23	**51.37**	+1127.86
Vicuna-7B	BI	41.63	**25.52**	+16.11
	Reverse*	**43.94**	71.37	-27.43

Table 3: WikiText-2 perplexity at 25% depth pruning. Lower is better. Gain denotes the perplexity reduction obtained by adding the LRA.

4.2 Main Results With Fine-Tuning

SHIFT-LLM itself requires no post-pruning fine-tuning; our primary setting is therefore the training-free evaluation in Sec. 4.1. We additionally combine the LRA with existing fine-tuning schemes only to assess its compatibility with gradient-based recovery. We consider LoRA fine-tuning, partial-layer fine-tuning (Lu et al., 2024), and integration with the state-of-the-art Navigation LLM pipeline.

LoRA Fine-tuning. Table 2 evaluates whether SHIFT-LLM remains complementary to LoRA-based post-pruning fine-tuning. Adding the LRA improves average zero-shot performance by **+1.41**, **+0.77**, **+1.14**, and **+0.72** points on Qwen2-1.5B, Qwen1.5-7B, Llama-3.1-8B-Instruct, and Vicuna-7B-v1.5, respectively. These results show that the explicit correction provided by the LRA remains complementary to parameter-efficient adaptation: the affine residual correction estimates the missing residual update, while fine-tuning further adapts the remaining model parameters.

Partial-layer Fine-tuning. Table 4 evaluates SHIFT-LLM with partial-layer fine-tuning under Reverse-order* pruning. Adding the LRA improves average zero-shot performance by **+0.98**, **+2.58**,

Table 4: Zero-shot results with fine-tuning on C4 dataset under Reverse-order* pruning. "Orig." denotes the unpruned model, "Partial" denotes standard post-pruning partial-layer fine-tuning, and "Partial+LRA" denotes partial-layer fine-tuning combined with the proposed LRA, whose affine residual correction is kept trainable during fine-tuning.

Dataset	Qwen2-1.5B			Qwen1.5-7B			Llama-3.1-8B-It			Vicuna-7B		
	Orig.	Partial	Partial +LRA	Orig.	Partial	Partial +LRA	Orig.	Partial	Partial +LRA	Orig.	Partial	Partial +LRA
BoolQ	73.36	**67.19**	65.78	81.90	**80.09**	79.69	83.98	**74.89**	63.67	80.55	**78.04**	77.77
PIQA	75.46	62.68	**62.95**	78.40	66.05	**67.36**	79.98	67.63	**70.02**	77.26	**68.44**	68.23
HellaSwag	65.41	47.81	**47.85**	76.97	58.13	**61.22**	79.27	62.30	**67.12**	73.76	60.03	**63.51**
WinoGrande	66.22	**54.70**	53.99	66.14	**58.72**	57.38	74.19	60.22	**64.01**	69.61	60.14	**62.19**
ARC-e	66.16	**39.27**	39.10	70.71	43.14	**46.89**	81.78	56.57	**64.44**	75.63	55.22	**59.01**
ARC-c	36.09	**28.33**	28.07	42.83	**34.73**	34.39	55.03	39.16	**44.80**	45.82	36.09	**39.25**
OBQA	36.40	**28.80**	27.00	41.80	31.60	**32.40**	43.00	35.60	**39.20**	45.00	35.40	**37.20**
Avg.	59.87	**46.97**	46.39	65.54	53.21	**54.19**	71.03	56.62	**59.20**	66.80	56.19	**58.16**

Table 5: Comparison with state-of-the-art pruning methods on Llama-3.1-8B-It using Alpaca dataset for fine-tuning. All previous results are taken from the Navigation LLM paper, and * indicates reproduced results reported in that work. "Navigation LLM+LRA" denotes Navigation LLM equipped with the proposed LRA.

Method	PR	BoolQ	PIQA	HSwag	OBQA	ARC-e	ARC-c	MMLU	CMMLU	Wino	Avg.
Original	0%	83.98	79.87	59.10	33.80	81.90	51.71	68.04	55.43	73.72	65.28
ShortGPT (BI)	25%	–	71.76	41.96	20.20	61.07	28.41	24.17	24.94	53.91	40.80
Short. LLaMA (PPL)	25%	–	76.28	49.31	26.40	72.90	38.05	33.67	27.24	57.93	47.72
Short. LLaMA (Taylor)	25%	–	71.38	49.64	27.40	68.48	41.81	28.61	25.04	71.35	47.96
SLEB (FT)	25%	–	75.73	49.73	26.80	69.70	38.65	43.05	33.38	63.85	50.11
LLM-Pruner*	20%	–	72.00	54.60	–	–	–	25.30	25.00	–	44.23
SliceGPT*	20%	–	68.30	47.50	–	–	–	28.80	24.80	–	42.35
LaCo*	20%	–	69.80	55.70	–	–	–	26.50	25.20	–	44.08
LLM-Streamline*	20%	–	71.50	61.10	–	–	–	45.50	29.40	–	51.88
GRASP*	20%	–	73.30	62.70	–	–	–	43.10	30.70	–	52.45
Navigation LLM*	25%	67.61	73.67	**53.02**	**30.80**	72.73	**46.08**	**66.28**	**55.31**	66.22	59.08
Navigation LLM+**LRA**	25%	**69.21**	**74.21**	52.08	29.00	**73.99**	45.56	65.94	54.96	**70.96**	**59.54**

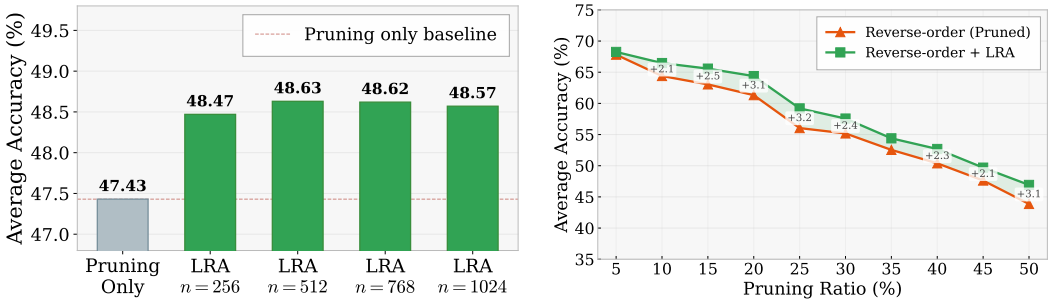
and **+1.97** points on Qwen1.5-7B, Llama-3.1-8B-Instruct, and Vicuna-7B-v1.5, respectively, while Qwen2-1.5B shows a small decrease of 0.58 points. Overall, the results show that SHIFT-LLM can provide additional recovery beyond partial-layer fine-tuning, with gains across most evaluated models.

Comparison with State-of-the-Art Pruning Methods. Navigation LLM (Lu et al., 2024) combines Reverse-order pruning with partial-layer fine-tuning and provides a strong depth-pruning baseline. Keeping its pruning strategy, fine-tuning protocol, and hyperparameters unchanged, we only add the proposed LRA. As shown in Table 5, this improves average accuracy from **59.08** to **59.54**, showing that adding the LRA does not degrade the strong fine-tuned baseline and can provide additional recovery.

4.3 Ablation Studies

We analyze three aspects of the proposed LRA: calibration set size, the importance of the residual parameterization, and robustness across pruning ratios. Additional ablations are provided in the supplementary material.

Effect of Calibration Set Size. Figure 3a evaluates calibration size under Magnitude-ℓ_1 pruning on Qwen2-1.5B at a pruning ratio of 25%. The proposed LRA improves over the pruning-only baseline for all calibration sizes, showing that its affine residual correction can be estimated effectively from limited calibration data. Since performance improves only slightly beyond 256 samples and quickly saturates, we use 256 calibration samples for all main experiments.



(a) Effect of calibration set size

(b) Effect of the pruning ratio

Figure 3: Ablation studies of the proposed LRA under different calibration sizes and pruning ratios.

Importance of the Residual Parameterization. We compare the proposed LRA with a Generic Affine baseline that directly fits $\hat{h}^{(\ell)} = h^{(\ell-1)}W^{(\ell)} + \mathbf{1}(b^{(\ell)})^\top$ using the same calibration data. Although both formulations have equivalent affine representational capacity through $W^{(\ell)} = I + A^{(\ell)}$, the LRA preserves the identity pathway and estimates only the missing residual update, providing a better residual inductive bias. Table 6 shows that this parameterization consistently performs better.

Criterion	Base	+Generic	+LRA
BI	45.78	36.96	**46.85**
Reverse*	43.71	37.99	**45.71**
Magnitude-ℓ_1	47.43	35.74	**48.47**

Table 6: LRA vs. Generic Affine on Qwen2-1.5B at 25% pruning. Average zero-shot accuracy over seven benchmarks.

Effect of the Pruning Ratio. Figure 3b evaluates Reverse-order pruning on Llama-3.1-8B-Instruct across different pruning ratios. Although performance decreases as more layers are removed, SHIFT-LLM consistently outperforms the pruning-only baseline, typically by **+2** to **+3** average accuracy points.

4.4 Computational Efficiency

As shown in Table 7, a rank-64 LRA, used in all experiments reported in this paper, requires less than 0.25% of the computation of a Llama-3.1-8B-Instruct block, with FLOPs excluding quadratic attention. Runtime is measured on a single V100 GPU. At 25% depth pruning, the LRA preserves a **1.27**× speedup over the original model while adding only a small runtime overhead to the pruned model. Exact merging further reduces latency to 44.9 ms, nearly matching the 44.7 ms pruned-only model. Thus, SHIFT-LLM provides accuracy recovery while largely preserving the inference-efficiency gains of depth pruning.

Per-block computational cost			
Component	Params	FLOPs/token	Relative Cost
Transformer block	218M	436M	100%
Full-rank LRA	16.8M	33.6M	7.7%
Rank-64 LRA	0.5M	1.0M	$< 0.25\%$
End-to-end runtime at 25% pruning			
Configuration	Params	Layers	Latency (ms)	Tokens/s
Original	8.03B	32	58.3 ± 0.3	21.4 ± 0.3
Pruned	6.29B	24	44.7 ± 0.3	27.7 ± 0.3
Pruned + rank-64 LRA	6.29B	32	46.0 ± 2.4	26.1 ± 1.2
Pruned + merged LRA	6.30B	25	44.9 ± 0.9	27.6 ± 0.3

Table 7: Computational efficiency of SHIFT-LLM on Llama-3.1-8B-Instruct.

5 Conclusion

We presented SHIFT-LLM, a training-free framework for mitigating the hidden-state distribution shift caused by depth pruning. Its Linear Residual Adapter (LRA) preserves the identity pathway and uses a closed-form affine residual correction to approximate the missing residual update. Experiments across models, pruning criteria, and evaluation settings show consistent performance recovery while preserving the efficiency benefits of depth pruning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with brainstorming and refining research ideas, provide feedback on research methodology and experimental design, and support the interpretation of experimental results. We also used generative AI tools for literature retrieval and discovery, drafting parts of the manuscript, editing and polishing the writing for clarity and readability, and assisting with the presentation and refinement of mathematical formulations. We did not use generative AI tools to generate synthetic datasets. All AI-assisted research ideas, mathematical formulations, interpretations, literature suggestions, and manuscript text were independently reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including all mathematical claims, text, results, and conclusions.

REFERENCES

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

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

Yi Cao, Wei-Jie Xu, Yucheng Shen, Weijie Shi, Chi-Min Chan, Jianfeng Qu, and Jiajie Xu. Pip: Perturbation-based iterative pruning for large language models. *arXiv preprint arXiv:2501.15278*, 2025.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 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.

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

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 1286–1305, 2021.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slimllm: Accurate structured pruning for large language models. *arXiv preprint arXiv:2505.22689*, 2025.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*, 2024.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

G Ling, Z Wang, Y Yan, and Q Liu. Slimgpt: layer-wise structured pruning for large language models (2024). *URL https://arxiv. org/abs/2412.18110*.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.

Kainan Liu, Yong Zhang, Ning Cheng, Zhitao Li, Shaojun Wang, and Jing Xiao. Grasp: Replace redundant layers with adaptive singular parameters for efficient model compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26344–26359, 2025.

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*, 2024.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 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, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 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.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Dmitriy Shopkhoev, Ammar Ali, Magauiya Zhussip, Valentin Malykh, Stamatios Lefkimmiatis, Nikos Komodakis, and Sergey Zagoruyko. Replaceme: Network simplification via depth pruning and transformer block linearization. *arXiv preprint arXiv:2505.02819*, 2025.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. A deeper look at depth pruning of llms. *arXiv preprint arXiv:2407.16286*, 2024.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6401–6417, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Experimental Setup and Pruning Criteria

A.1 Experimental Setup

We evaluate SHIFT-LLM on five decoder-only LLMs: Qwen2-1.5B, Qwen1.5-7B, Qwen2.5-14B, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Vicuna-7B-v1.5 (Zheng et al., 2023). Unless otherwise stated, all SHIFT-LLM experiments use rank-64 LRAs. For data-driven pruning criteria (e.g., BI, Taylor, and PPL), we use 2,000 samples from the C4 validation split (Dodge et al., 2021), while the affine residual corrections of the LRAs are fitted using 256 samples from the same split.

SHIFT-LLM itself requires no post-pruning fine-tuning. For the additional compatibility experiments with LoRA and partial-layer fine-tuning, we train on 10,000 samples from the C4 training split and validate on 200 samples from the C4 validation split, using a maximum sequence length of 512, a batch size of 25, a learning rate of 3×10^{-5}, and the standard causal language modeling objective. In partial-layer fine-tuning, the language modeling head and the last three unpruned layers are fine-tuned. For comparison with Navigation LLM (Lu et al., 2024), we follow its protocol and fine-tune on the full Alpaca-Cleaned dataset (Taori et al., 2023) ($\sim$52K instruction-response pairs) with a maximum sequence length of 256, a learning rate of 1×10^{-5}, an effective batch size of 64, and 2 training epochs.

For zero-shot evaluation, we report results on seven standard benchmarks: BoolQ Clark et al. (2019), PIQA Bisk et al. (2020), HellaSwag Zellers et al. (2019), WinoGrande Sakaguchi et al. (2021), ARC-Easy Clark et al. (2018), ARC-Challenge Clark et al. (2018), and OpenBookQA Mihaylov et al. (2018). These tasks cover yes/no question answering, physical commonsense reasoning, sentence completion, coreference resolution, and science question answering. Following prior pruning work, we adopt a likelihood-based evaluation protocol. For BoolQ, we compare the log-likelihood of the continuations "yes" and "no" given the input context. For the remaining multiple-choice benchmarks, we compute the conditional log-probability of each candidate answer and select the highest-scoring option. For datasets with multi-token answer choices, namely HellaSwag, ARC-Challenge, and OpenBookQA, we additionally report length-normalized accuracy to reduce the bias toward shorter completions. All evaluations are performed on the full benchmark splits without subsampling. Importantly, none of the evaluation datasets is used for calibration or fine-tuning, ensuring a strict separation between adaptation and evaluation data. All experiments are conducted on NVIDIA V100 GPUs.

A.2 Layer Pruning Criteria

To evaluate the proposed method under diverse pruning settings, we consider several widely used criteria for selecting layers to remove. These criteria range from simple heuristics to data-dependent importance measures, allowing us to test the proposed LRA-based correction across a broad spectrum of pruning patterns.

Specifically, we consider six main layer selection strategies: *Reverse-order* (Men et al., 2025), *Magnitude-ℓ_1* (Li et al., 2016; Kim et al., 2024; Lu et al., 2024), *Magnitude-ℓ_2* (Li et al., 2016; Kim et al., 2024; Lu et al., 2024), *Taylor* (Lu et al., 2024), *Perplexity-based* (PPL) (Lu et al., 2024), and *Block Influence* (BI) (Men et al., 2025). The *Reverse-order* strategy removes deeper layers first based on their position in the network. We also evaluate *Reverse-order**, a modified variant that preserves the final Transformer layer, since its output feeds directly into the prediction head and cannot be corrected by a subsequent LRA. The two *Magnitude*-based criteria rank layers according to the norms of their weights, using either the ℓ_1 or ℓ_2 norm. The *Taylor* criterion estimates layer importance using a first-order approximation of the loss change on a calibration set. The *PPL* criterion measures the increase in perplexity caused by removing a single layer and treats layers with smaller perplexity degradation as less important. Finally, *Block Influence* measures how much a layer changes its input representation, using the average cosine similarity between the layer input and output as a proxy for redundancy.

Together, these criteria provide diverse pruning patterns for evaluating the robustness and generality of SHIFT-LLM across different layer-selection strategies.

Table 8: Per-benchmark zero-shot accuracy corresponding to the average results reported in Table 1. Results are shown for different pruning criteria at a pruning ratio of 25%, without post-pruning fine-tuning. "+LRA" denotes the pruned model equipped with the proposed LRA.

Model	Method	PR	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
Qwen2-1.5B	Original	0%	73.36	75.46	65.41	66.22	66.16	36.09	36.40	59.87
Qwen1.5-7B	Original	0%	81.90	78.40	76.97	66.14	70.71	42.83	41.80	65.54
Llama-3.1-8B-It	Original	0%	83.98	79.98	79.27	74.19	81.78	55.03	43.00	71.03
Vicuna-7B	Original	0%	80.55	77.26	73.76	69.61	75.63	45.82	45.00	66.80
Qwen2-1.5B	ShortGPT (BI)	25%	55.57	62.51	43.17	55.01	45.20	**28.58**	**30.40**	45.78
Qwen2-1.5B	ShortGPT (BI)+LRA	25%	**57.74**	**64.80**	**44.88**	**56.12**	**48.32**	26.71	29.40	**46.85**
Qwen1.5-7B	ShortGPT (BI)	25%	74.40	65.83	**56.62**	60.06	43.43	**32.08**	**32.00**	52.06
Qwen1.5-7B	ShortGPT (BI)+LRA	25%	**76.54**	**67.03**	55.54	**63.93**	**45.50**	32.00	29.20	**52.82**
Llama-3.1-8B-It	ShortGPT (BI)	25%	77.74	68.23	60.62	66.46	54.80	**40.78**	31.60	57.18
Llama-3.1-8B-It	ShortGPT (BI)+LRA	25%	**82.14**	**70.84**	**66.52**	**71.82**	**64.60**	40.61	**34.80**	**61.62**
Vicuna-7B	ShortGPT (BI)	25%	64.07	66.65	**59.95**	66.38	56.65	**35.24**	**36.40**	55.05
Vicuna-7B	ShortGPT (BI)+LRA	25%	**69.63**	**66.92**	56.63	**66.85**	**58.63**	34.81	35.20	**55.52**
Qwen2-1.5B	Reverse-order*	25%	62.84	58.81	40.00	56.75	30.98	26.19	**30.40**	43.71
Qwen2-1.5B	Reverse-order*+LRA	25%	**64.77**	**62.08**	**41.25**	**59.98**	**36.20**	**26.88**	28.80	**45.71**
Qwen1.5-7B	Reverse-order*	25%	63.33	65.34	51.05	56.59	43.47	31.40	**33.60**	49.25
Qwen1.5-7B	Reverse-order*+LRA	25%	**79.27**	**67.52**	**53.57**	**63.30**	**47.26**	**33.62**	32.00	**53.79**
Llama-3.1-8B-It	Reverse-order*	25%	62.23	59.85	27.58	55.72	36.03	30.29	31.60	43.33
Llama-3.1-8B-It	Reverse-order*+LRA	25%	**62.42**	**70.84**	**64.26**	**71.82**	**64.23**	**44.11**	**35.80**	**59.07**
Vicuna-7B	Reverse-order*	25%	63.27	**66.16**	**53.76**	62.90	**55.51**	**35.32**	**37.40**	**53.47**
Vicuna-7B	Reverse-order*+LRA	25%	**68.56**	64.36	51.86	**64.33**	53.32	34.47	33.80	52.96
Qwen2-1.5B	Magnitude-ℓ_1	25%	61.74	65.83	43.02	51.93	51.89	**26.45**	31.20	47.44
Qwen2-1.5B	Magnitude-ℓ_1+LRA	25%	**62.29**	**67.68**	**45.37**	**53.99**	**52.57**	25.60	**31.80**	**48.47**
Qwen1.5-7B	Magnitude-ℓ_1	25%	56.02	54.57	34.77	**51.70**	35.14	26.02	26.80	40.72
Qwen1.5-7B	Magnitude-ℓ_1+LRA	25%	**61.59**	**66.16**	**40.39**	51.54	**55.56**	**26.11**	**31.80**	**47.59**
Llama-3.1-8B-It	Magnitude-ℓ_1	25%	**54.83**	55.43	27.09	48.62	25.17	**25.51**	**28.00**	37.81
Llama-3.1-8B-It	Magnitude-ℓ_1+LRA	25%	49.82	**59.58**	**30.10**	**51.70**	**37.04**	22.10	25.60	**39.42**
Vicuna-7B	Magnitude-ℓ_1	25%	37.83	52.34	**26.43**	47.99	**26.01**	**27.05**	**26.40**	34.86
Vicuna-7B	Magnitude-ℓ_1+LRA	25%	**39.05**	**53.05**	26.14	**49.96**	25.34	26.88	25.20	**35.09**
Qwen2-1.5B	Magnitude-ℓ_2	25%	61.47	62.73	37.04	50.99	43.39	23.29	27.40	43.76
Qwen2-1.5B	Magnitude-ℓ_2+LRA	25%	**62.05**	**64.58**	**41.04**	**51.70**	**45.79**	**23.89**	**31.40**	**45.78**
Qwen1.5-7B	Magnitude-ℓ_2	25%	60.52	60.83	36.69	50.51	44.49	24.57	**29.20**	43.83
Qwen1.5-7B	Magnitude-ℓ_2+LRA	25%	**61.96**	**64.36**	**36.90**	**52.17**	**49.33**	**26.88**	28.60	**45.74**
Llama-3.1-8B-It	Magnitude-ℓ_2	25%	45.57	51.80	27.08	49.72	26.05	**25.00**	**26.80**	36.00
Llama-3.1-8B-It	Magnitude-ℓ_2+LRA	25%	**53.00**	**59.09**	**29.02**	**51.78**	**37.63**	21.16	25.40	**39.58**
Vicuna-7B	Magnitude-ℓ_2	25%	37.92	**52.77**	**26.04**	48.86	**25.13**	**28.41**	**26.60**	35.10
Vicuna-7B	Magnitude-ℓ_2+LRA	25%	**54.74**	52.45	25.80	**50.36**	24.96	26.28	24.60	**37.03**
Qwen2-1.5B	Taylor	25%	62.72	62.40	42.02	57.85	38.76	**27.99**	**29.80**	45.93
Qwen2-1.5B	Taylor+LRA	25%	**66.54**	**63.60**	**42.46**	**59.83**	**40.95**	26.79	28.20	**46.91**
Qwen1.5-7B	Taylor	25%	54.53	69.21	**58.95**	58.64	53.20	**30.97**	32.80	51.19
Qwen1.5-7B	Taylor+LRA	25%	**62.35**	**70.40**	57.33	**59.27**	**57.53**	30.80	**33.60**	**53.04**
Llama-3.1-8B-It	Taylor	25%	62.23	59.74	27.56	55.80	35.98	30.20	31.60	43.30
Llama-3.1-8B-It	Taylor+LRA	25%	**62.42**	**70.73**	**64.21**	**71.67**	**64.18**	**44.11**	**35.80**	**59.02**
Vicuna-7B	Taylor	25%	77.68	**65.45**	**57.79**	67.01	**56.86**	**37.03**	34.80	56.66
Vicuna-7B	Taylor+LRA	25%	**79.82**	**65.45**	57.70	**67.40**	56.69	34.64	**35.00**	**56.67**
Qwen2-1.5B	PPL	25%	61.31	63.60	41.72	54.46	42.34	**27.90**	28.8	45.73
Qwen2-1.5B	PPL+LRA	25%	**62.17**	**65.23**	**43.23**	**58.72**	**48.70**	26.88	**29.40**	**47.76**
Qwen1.5-7B	PPL	25%	38.53	54.03	27.31	49.33	26.39	**25.85**	**29.8**	35.89
Qwen1.5-7B	PPL+LRA	25%	**47.80**	**56.75**	**29.87**	**50.75**	**30.89**	22.44	24.2	**37.53**
Llama-3.1-8B-It	PPL	25%	80.09	72.36	65.56	70.17	63.17	**41.21**	35.6	61.17
Llama-3.1-8B-It	PPL+LRA	25%	**81.71**	**73.67**	**66.66**	**70.80**	**65.99**	40.87	**35.8**	**62.21**
Vicuna-7B	PPL	25%	**75.38**	**65.45**	**57.48**	**64.64**	**55.01**	**34.13**	**33.80**	**55.13**
Vicuna-7B	PPL+LRA	25%	72.35	62.13	50.64	61.40	47.22	31.40	30.40	50.79

A.3 Per-Benchmark Results Without Fine-Tuning

Table 8 provides the per-benchmark zero-shot results corresponding to the average scores reported in Table 1. Specifically, we report accuracy on BoolQ, PIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA for each pruning criterion at a pruning ratio of 25%, without any post-pruning fine-tuning. The "Original" row denotes the unpruned model, while each pruning

Table 9: Zero-shot results on Qwen2.5-14B at 25% depth pruning.

Method	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
Original	86.27	81.07	82.91	75.14	82.45	58.87	45.20	73.13
BI	66.51	67.03	54.66	63.30	50.97	36.69	**34.20**	53.34
BI + LRA	**70.52**	**67.46**	**59.20**	**71.35**	**56.06**	**37.46**	32.20	**56.32**
Reverse*	62.23	59.19	42.48	58.01	40.32	31.83	**33.00**	46.72
Reverse* + LRA	**64.92**	**66.92**	**51.90**	**66.61**	**48.57**	**34.90**	30.80	**52.08**

Table 10: Effect of low-rank factorization of the affine residual correction under Magnitude-ℓ_1 pruning on Qwen2-1.5B at 25% depth pruning.

LRA Configuration	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
Full-rank	62.23	67.95	43.25	54.38	52.99	25.60	32.20	48.38
Rank 64	62.29	67.68	45.37	53.99	52.57	25.60	31.80	48.47
Rank 128	62.26	68.44	45.65	52.96	53.66	25.51	32.20	**48.67**
Rank 256	62.26	69.04	45.18	52.57	53.79	25.51	31.80	48.59
Rank 512	62.26	68.39	44.21	53.59	53.87	25.34	32.20	48.55

criterion is reported both before and after adding the proposed LRA. These detailed results show how the average improvements in the main paper are distributed across individual benchmarks.

A.4 Results on Qwen2.5-14B

To further evaluate scalability, we apply SHIFT-LLM to Qwen2.5-14B at 25% depth pruning. Table 9 reports the full per-benchmark results under Block Influence and Reverse-order* pruning. The proposed LRA improves average accuracy by **+2.98** points under BI and **+5.36** points under Reverse-order*, showing that the training-free correction remains effective at the 14B scale.

B Additional Ablation Studies

B.1 Effect of Low-Rank Factorization

Table 10 examines the effect of low-rank factorization on the affine residual correction of the proposed LRA under Magnitude-ℓ_1 pruning on Qwen2-1.5B at 25% depth pruning. Performance is reported as the average zero-shot accuracy over seven benchmarks. The low-rank variants match or slightly outperform the full-rank LRA, suggesting that the fitted correction contains directions that can be removed without degrading its ability to approximate the missing residual update. Rank 128 achieves the highest accuracy, while rank 64 provides a stronger efficiency–performance trade-off and is therefore used in all reported experiments unless otherwise stated.

Table 11: Sensitivity to the calibration domain on Vicuna-7B at 25% depth pruning. Results are average zero-shot accuracy over seven benchmarks.

Criterion	Pruned	+LRA (C4)	+LRA (WikiText)	+LRA (Alpaca)
BI	55.05	55.66	56.14	54.66
Reverse*	53.47	52.96	53.36	52.20
Magnitude-ℓ_1	34.86	35.09	35.80	35.17
Magnitude-ℓ_2	35.10	37.02	38.16	37.32

Table 12: Random pruning with and without the proposed LRA on Qwen2-1.5B at 25% depth pruning.

Method	BoolQ	PIQA	HSwag	Wino	ARC-e	ARC-c	OBQA	Avg.
Random	57.83	55.82	27.42	50.59	28.41	22.87	24.60	38.22
Random + LRA	38.84	54.84	27.60	48.70	31.06	22.87	28.40	36.07

Table 13: Independent vs. sequential LRA calibration on Llama-3.1-8B-Instruct at 25% depth pruning.

Criterion	Calibration	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
BI	Independent	82.14	70.84	46.21	71.82	64.60	38.14	22.60	**56.62**
BI	Sequential	83.21	70.13	45.53	72.30	64.06	37.03	21.80	56.29
Reverse*	Independent	62.42	70.84	46.12	71.82	64.23	41.21	25.60	54.61
Reverse*	Sequential	62.54	71.87	46.15	72.61	66.62	41.21	24.20	**55.03**

B.2 Calibration-Domain Sensitivity

To determine whether the weaker results on Vicuna-7B are caused by a mismatch between the C4 calibration data and the model's instruction-tuning distribution, we fit the LRA using C4, WikiText, and Alpaca calibration samples. Table 11 shows that changing the calibration domain does not eliminate the negative gain under Reverse-order* pruning, and differences across calibration sources remain relatively small in most settings. In particular, instruction-aligned Alpaca calibration does not consistently outperform C4. These results indicate that the weaker Vicuna results are not primarily caused by calibration-domain mismatch.

B.3 Random Layer Selection and Failure Mode

We further examine when the token-wise affine correction fails by applying the LRA after random layer removal. Unlike principled pruning criteria, random selection may remove blocks whose missing residual updates contain stronger nonlinear or cross-token structure. As shown in Table 12, the LRA decreases average accuracy from 38.22 to 36.07 in this setting. This result supports the interpretation that SHIFT-LLM is most effective when the pruning criterion selects blocks whose missing residual updates are sufficiently amenable to low-complexity approximation.

B.4 Independent vs. Sequential Calibration

Our default procedure estimates each LRA independently from activations of the original model. To test whether accumulated corrections create a calibration–inference mismatch, we also perform sequential calibration, where each fitted LRA is inserted before collecting activations for the next pruning site. Table 13 shows nearly identical performance between the two strategies. This indicates that the corrected hidden states remain sufficiently close to the original representations for independent closed-form calibration to be effective.

B.5 Robustness to Calibration Samples

Because the LRA is estimated in closed form, identical calibration data produces identical parameters and no optimization randomness is involved. We therefore test sensitivity to the composition of the calibration set using two random calibration seeds on Llama-3.1-8B-Instruct. The resulting gains remain consistent: BI improves by +4.44 and +5.09 points, while Reverse-order* improves by +15.74 and +16.21 points. The variation remains below one point, indicating that the training-free correction is robust to the sampled calibration examples.

B.6 Training vs. Freezing the LRAs During Fine-Tuning

Although SHIFT-LLM itself is training-free, we additionally examine whether its closed-form solution can serve as a useful initialization when post-pruning fine-tuning is available. Table 14 compares freezing the fitted LRAs with allowing their affine residual corrections to remain trainable during LoRA fine-tuning. Training the correction consistently provides additional gains, indicating that the closed-form solution can be further refined when gradient-based recovery is desired.

B.7 Effect of the Reverse Pruning Variant

We compare the original Reverse-order strategy with Reverse-order* on Qwen2-1.5B. Reverse-order removes the deepest layers directly, whereas Reverse-order* preserves the final Transformer block and instead removes the preceding deep layers. Under Reverse-order*, the proposed LRA improves average accuracy from 43.71 to 45.71 (**+2.00**), whereas under the original Reverse-order strategy the accuracy decreases slightly from 44.36 to 43.80 (-0.56). Preserving the final block retains the pretrained transformation immediately preceding the language-modeling head, while the LRA corrects earlier missing residual updates. We therefore use Reverse-order* in the main experiments.

B.8 Comparison with a Closely Related Linear Recovery Baseline

Among training-free depth-pruning methods, ReplaceMe (Shopkhoev et al., 2025) is closely related to SHIFT-LLM: both use a small calibration set to estimate a lightweight linear correction after block removal, without gradient-based training. However, the two methods differ in both the representation used for regression and the quantity directly approximated.

ReplaceMe considers a retained Transformer block i, whose post-attention representation and MLP contribution are denoted by Y_i and M_i, respectively. After removing the subsequent blocks $i+1, \ldots, i+n$, it estimates a linear transformation T by solving

$$T^* = \arg\min_T \|M_i T + Y_i - L_{i+n}\|_F^2 , \tag{15}$$

where L_{i+n} is the original hidden state after the removed block sequence. Thus, ReplaceMe transforms the MLP contribution M_i of the retained predecessor block, while Y_i remains a fixed additive term, so that their sum directly approximates the original output after the removed span. This matches the formulation in ReplaceMe, where the learned transformation is applied to the predecessor MLP output and the blocks $i+1, \ldots, i+n$ are bypassed.

SHIFT-LLM instead performs a local correction at each pruning site. For a removed block ℓ, the original Transformer block computes

$$h_{\text{out}}^{(\ell)} = h_{\text{in}}^{(\ell)} + \Delta_\ell, \qquad \Delta_\ell = f_\ell\Big(h_{\text{in}}^{(\ell)}\Big) , \tag{16}$$

where Δ_ℓ denotes the missing residual update introduced by removing block ℓ. Rather than directly regressing the full output hidden state, SHIFT-LLM estimates only this missing residual update using the full hidden state at the pruning site:

$$(A^*, b^*) = \arg\min_{A,b} \left\| h_{\text{in}}^{(\ell)} A + \mathbf{1} b^\top - \Delta_\ell \right\|_F^2 + \lambda \|A\|_F^2 . \tag{17}$$

The resulting LRA is

$$\hat{h}_{\text{out}}^{(\ell)} = h_{\text{in}}^{(\ell)} + h_{\text{in}}^{(\ell)} A^* + \mathbf{1}(b^*)^\top . \tag{18}$$

Table 14: Frozen vs. trainable LRAs during LoRA fine-tuning under Reverse-order* pruning at 25% depth pruning.

Model	Frozen	Trainable	Gain
Qwen2-1.5B	47.98	**48.94**	+0.96
Qwen1.5-7B	56.08	**56.28**	+0.20
Llama-3.1-8B-It	60.32	**61.12**	+0.80

Table 15: Comparison with ReplaceMe (L2) at 25% depth pruning. Results are average zero-shot accuracy over seven benchmarks using the same 256-sample C4 calibration budget.

Model	Criterion	Pruned	ReplaceMe	SHIFT-LLM
Qwen2-1.5B	BI	45.8	43.9 (−1.9)	**46.9** (+1.1)
Qwen2-1.5B	Reverse*	43.7	44.3 (+0.6)	**45.7** (+2.0)
Qwen1.5-7B	BI	52.06	50.34 (−1.72)	**52.82** (+0.76)
Qwen1.5-7B	Reverse*	49.25	53.62 (+4.37)	**53.79** (+4.54)
Llama-3.1-8B	BI	57.18	59.46 (+2.28)	**61.62** (+4.44)
Llama-3.1-8B	Reverse*	43.33	58.02 (+14.69)	**59.07** (+15.74)
Vicuna-7B	BI	55.05	54.33 (−0.72)	**55.52** (+0.48)
Vicuna-7B	Reverse*	53.47	46.40 (−7.07)	**52.95** (−0.52)

The two methods therefore differ in two important ways. First, ReplaceMe learns its transformation from the MLP contribution M_i of the retained predecessor block, whereas SHIFT-LLM uses the full hidden state $h_{\text{in}}^{(\ell)}$ at the pruning site as the input to its affine residual correction. Second, ReplaceMe directly fits the full hidden state L_{i+n} after the removed span through $Y_i + M_i T$, whereas SHIFT-LLM explicitly preserves $h_{\text{in}}^{(\ell)}$ as the identity pathway and estimates only the missing residual update Δ_ℓ. The affine bias b^* additionally allows SHIFT-LLM to capture a constant offset in the missing residual update.

These design choices provide a more direct local approximation of each removed block's missing residual update: the correction is conditioned on the full hidden state that originally drives f_ℓ, while the identity pathway is preserved explicitly and only the missing residual update is estimated.

The importance of the identity-plus-correction parameterization is also supported by the Generic Affine ablation in Table 6. Although a generic affine mapping has the same representational capacity as the proposed parameterization, it performs substantially worse in our experiments. This suggests that explicitly preserving the identity pathway and estimating only the missing residual update provides a beneficial parameterization for post-pruning recovery.

Table 15 provides a direct comparison using the same 256-sample C4 calibration budget. Across all evaluated model–pruning combinations, SHIFT-LLM achieves higher average zero-shot accuracy than ReplaceMe (L2). The difference is particularly large on Vicuna-7B under Reverse-order* pruning, where ReplaceMe substantially degrades the pruning-only model, while SHIFT-LLM produces a considerably smaller degradation.

C Additional Computational Analysis

As reported in Table 7 of the main paper, SHIFT-LLM introduces only a small computational overhead relative to the removed Transformer blocks. Here, we provide additional details on how this overhead scales with the number of pruned layers.

For a Llama-3.1-8B-Instruct block, the rank-64 LRA requires approximately 0.5M parameters and 1.0M FLOPs per token, compared with 218M parameters and 436M FLOPs per token for the original Transformer block, excluding the quadratic attention term. More generally, for a pruning ratio ρ, depth pruning removes $\lfloor \rho L \rfloor$ Transformer blocks, while the LRA overhead grows only linearly with the number of pruning sites.

At 25% pruning of the 32-layer Llama-3.1-8B-Instruct model, eight Transformer blocks are removed, corresponding to approximately 1.74B parameters and 3.49G FLOPs per token. Replacing the eight pruning sites with rank-64 LRAs adds back only approximately 4.2M parameters and 8.4M FLOPs per token, i.e., less than 0.25% of the removed computational cost. When multiple pruned layers are consecutive, their affine mappings can additionally be merged exactly into a single transformation, as described in the main paper.

C.1 Runtime Measurement Details

Runtime measurements reported in Table 7 of the main paper are obtained on a single NVIDIA V100 GPU using an input length of 128 tokens and greedy generation of 128 output tokens. The reported latency and throughput values are measured for the original model, the depth-pruned model, the pruned model with rank-64 LRAs, and the exactly merged configuration.