PRISM: POLICY-DRIVEN RETRAINING-FREE INTER/INTRA-LAYER STRUCTURED MERGING FOR ACCURATE AND EFFICIENT LLM PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) incur substantial computation, memory, and latency costs during inference, motivating structured pruning for efficient deployment. However, directly removing layers or channels discards learned representations, often requiring fine-tuning or reconstruction to recover accuracy. Inter-layer pruning offers substantial computational savings at the risk of severe accuracy degradation, whereas intra-layer pruning provides finer-grained compression with more limited end-to-end acceleration. We introduce *PRISM*, a policy-driven, retraining-free framework that unifies inter- and intra-layer structured compression through weight merging. Rather than simply deleting redundant structures, *PRISM* transfers their information into retained weights, mitigating representation loss without additional training. For inter-layer compression, MHA-Removed FFN Composition removes selected multi-head attention (MHA) layers and composes adjacent feed-forward network (FFN) layers to exploit redundancy across layers. For intra-layer compression, Structure-Aware Channel Reconstruction merges channels while accounting for the distinct structural characteristics of MHA and FFN modules. To coordinate both compression levels, an Activation Distortion Score captures directional and magnitude changes in activations and guides the joint selection of inter-layer merging locations and layer-wise intra-layer sparsity. At 30% sparsity, *PRISM* achieves 1.25% lower perplexity than a state-of-the-art structured pruning baseline, while reducing inference computation and memory usage to 72.9% and 70.5% of the dense baseline, respectively. These results demonstrate that combining structured information transfer with activation-aware sparsity allocation improves the accuracy–efficiency trade-off of LLM compression without retraining.

1 INTRODUCTION

Recent large language models (LLMs) (Achiam et al., 2023; Touvron et al., 2023b) have achieved remarkable performance across a wide range of natural language understanding and generation tasks, and are rapidly being deployed in real-world services. However, as model size increases, inference requires substantially higher computation, memory footprint, and latency, which becomes particularly restrictive in on-device scenarios where privacy preservation and low response latency are critical. Therefore, effective model compression is essential for deploying LLMs under limited hardware resources while preserving model quality. Among various LLM compression techniques (Kim et al., 2026; Lee et al., 2026), pruning is a representative approach that reduces computation and memory usage by removing unnecessary or redundant weights.

Pruning methods can be broadly categorized into unstructured pruning (Frantar & Alistarh, 2023; Sun et al., 2023) and structured pruning (Ma et al., 2023; Ashkboos et al., 2024). Unstructured pruning can achieve high sparsity, but its irregular sparsity patterns are difficult to translate into practical speedup on general hardware. In contrast, structured pruning removes hardware-friendly units such as channels, layers, or blocks, simplifying the computation graph and reducing both memory access and arithmetic operations, thereby improving end-to-end inference latency. Despite these advantages, most existing structured pruning methods (Ma et al., 2023; Men et al., 2025) directly remove structural units, inevitably discarding the representations learned by those structures. Such removal perturbs the activations passed to subsequent layers, and the resulting errors can accumulate across layers

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.

in LLMs, leading to quality degradation. Consequently, post-pruning fine-tuning or reconstruction is typically required to recover the altered representations. However, fine-tuning incurs substantial memory overhead due to backpropagation and optimizer states (*e.g.*, LLM-Pruner (Ma et al., 2023) requires more than 32GB GPU memory for pruning a 7B model), while reconstruction also introduces non-negligible computation cost due to large-scale weight optimization (***Lim. 1***).

Structured pruning can be further divided by granularity into inter-layer methods (Wang et al., 2025) and intra-layer methods (He & Lin, 2025). Inter-layer methods remove coarse-grained structures such as Transformer (Vaswani et al., 2017) blocks or layers, substantially simplifying the computation graph and offering large latency gains. However, they often induce severe activation shifts and quality degradation. In contrast, intra-layer methods adjust fine-grained structures such as channels or hidden dimensions within each layer, better preserving model quality but providing limited latency improvement because the overall computation graph remains largely unchanged. Therefore, existing single-granularity methods (Wang et al., 2025; He & Lin, 2025) cannot fully exploit the heterogeneous redundancy and sensitivity across both layer-level and channel-level structures in LLMs (***Lim. 2***).

To address this issue, recent studies have explored hybrid-granularity structured pruning (Xia et al., 2024; Chen et al., 2024), which combines the complementary benefits of inter-/intra-layer compression. By jointly considering coarse-grained removal for computation reduction and fine-grained adjustment for quality preservation, hybrid-granularity pruning can provide a more flexible accuracy-efficiency trade-off than single-granularity approaches. However, existing hybrid methods still lack a systematic policy for determining the compression ratio across granularities and the layer-wise sparsity allocation. Instead, they often rely on predefined pruning schedules or large-scale fine-tuning for recovery. For example, Sheared-LLaMA (Xia et al., 2024) and LEMON (Chen et al., 2024) compress various structures through removal or merging, but require approximately 50B training tokens to recover the resulting quality loss. Thus, existing hybrid approaches lack a principled policy for effectively realizing hybrid-granularity compression (***Lim. 3***).

To overcome these limitations, we propose *PRISM*, a policy-driven, retraining-free inter-/intra-layer structured merging framework. *PRISM* first uses an Activation Distortion Score (ADS)-based Sparsity Allocation Policy (SAP) to jointly determine the locations for inter-layer merging and the layer-wise sparsity for intra-layer merging. At the inter-layer level, *PRISM* applies *MHA-Removed FFN Composition* (MRFC), which removes selected multi-head attention (MHA) layers and composes adjacent feed-forward network (FFN) layers, achieving coarse-grained computation reduction while mitigating representation loss. At the intra-layer level, *PRISM* employs Structure-Aware Channel Reconstruction (SACR) to merge redundant channels inside MHA and FFN layers while respecting their structural characteristics. Experimental results show that, at a sparsity ratio of 30%, *PRISM* improves perplexity (PPL) by 1.25% over the state-of-the-art structured pruning method (He & Lin, 2025), while reducing inference computation and memory usage to 72.9% and 70.5% of the dense baseline, respectively. Our main contributions are summarized as follows:

- ***Contribution 1***) We propose an ADS-based SAP to address the non-principled policy selection problem in existing hybrid-granularity methods. SAP evaluates the activation distortion of candidate merging operations and jointly determines inter-layer merging locations and layer-wise intra-layer sparsity, enabling principled hybrid merging without additional fine-tuning (solution for ***Lim. 3***).

- ***Contribution 2***) We reformulate structured pruning as *structured weight merging* rather than direct removal to reduce representation loss caused by structural compression. *PRISM* transfers the information of removed structures into merged structures, reducing compression-induced information loss without gradient-based retraining (solution for ***Lim. 1***).

- ***Contribution 3***) We propose a hybrid structured merging strategy consisting of *MRFC* and *SACR* to jointly exploit inter-layer and intra-layer redundancy. MRFC removes selected MHA layers and composes adjacent FFN layers for coarse-grained computation reduction, while SACR merges redundant channels inside MHA and FFN layers in a structure-aware manner to mitigate quality degradation (solution for ***Lim. 2***).

2 BACKGROUND

2.1 TRANSFORMER-BASED LLM COMPUTATION

Recent LLMs typically adopt a decoder-only architecture composed of stacked Transformer blocks, each consisting of a MHA layer and a FFN layer. Let n denote the input sequence length, d the hidden dimension, H the number of attention heads, d_h the head dimension, d_f the FFN hidden dimension, and L the number of blocks. The computation of the l-th block with residual connections is written as

$$\mathbf{X_F}^{(l)} = \mathrm{MHA}^{(l)}\Big(\mathbf{X_A}^{(l)}\Big) + \mathbf{X_A}^{(l)}, \quad \mathbf{X_A}^{(l+1)} = \mathrm{FFN}^{(l)}\Big(\mathbf{X_F}^{(l)}\Big) + \mathbf{X_F}^{(l)}, \tag{1}$$

where $\mathbf{X_A}^{(l)}, \mathbf{X_F}^{(l)} \in \mathbb{R}^{n\times d}$ denote the input activations of the MHA and FFN layers, respectively. The MHA layer first projects the input activation into query, key, and value representations for each head:

$$\mathbf{Q}_i = \mathbf{X_A}\mathbf{W}_{q,i}, \quad \mathbf{K}_i = \mathbf{X_A}\mathbf{W}_{k,i}, \quad \mathbf{V}_i = \mathbf{X_A}\mathbf{W}_{v,i}, \tag{2}$$

where $i \in \{1, \ldots, H\}$ is the head index, and $\mathbf{W}_{q,i}, \mathbf{W}_{k,i}, \mathbf{W}_{v,i} \in \mathbb{R}^{d\times d_h}$ are the projection weights of the i-th head. Each head computes the attention score using $\mathbf{Q}_i\mathbf{K}_i^\top/\sqrt{d_h}$, applies softmax to obtain attention probabilities, and multiplies them with $\mathbf{V}_i$ to generate the head-wise attention output. The output of each head is then projected back to the hidden dimension by $\mathbf{W}_{o,i}$, and all head outputs are aggregated to form the final MHA output. The FFN layer further transforms the token-wise representation in a channel-wise manner as

$$\mathbf{G} = \mathbf{X_F}\mathbf{W}_g, \quad \mathbf{U} = \mathbf{X_F}\mathbf{W}_u, \quad \mathbf{Z} = \sigma(\mathbf{G}) \odot \mathbf{U}, \quad \mathrm{FFN}(\mathbf{X_F}) = \mathbf{Z}\mathbf{W}_d, \tag{3}$$

where $\mathbf{W}_g, \mathbf{W}_u \in \mathbb{R}^{d\times d_f}$ and $\mathbf{W}_d \in \mathbb{R}^{d_f\times d}$ are projection weights, $\mathbf{G}, \mathbf{U}, \mathbf{Z} \in \mathbb{R}^{n\times d_f}$ are intermediate activations, $\odot$ denotes the Hadamard product, and $\sigma(\cdot)$ denotes the SiLU (Elfwing et al., 2018) activation function.

2.2 STRUCTURED PRUNING FOR LLMS

Structured pruning is a model compression technique that simplifies the computation graph by removing weights or operations in structured units such as channels, layers, or blocks. Unlike unstructured pruning, it preserves regular computation patterns, making it effective for reducing inference latency while maintaining parallelism and memory-access efficiency on real hardware. In Transformer-based LLMs, the dominant computation comes from MHA and FFN layers, with an overall complexity of $O(L(nd^2 + n^2d))$. Here, nd^2 corresponds to channel-wise linear projections in Q/K/V/O and FFN layers, while n^2d corresponds to token-wise attention score computation and value aggregation. Structured pruning reduces these costs differently depending on the pruning granularity.

Inter-layer Pruning. Inter-layer pruning removes coarse-grained structures such as Transformer blocks or MHA/FFN layers, thereby reducing the effective number of layers L. FinerCut (Zhang et al., 2024), LaCo (Yang et al., 2024), ShortGPT (Men et al., 2025), and SLEB (Song et al., 2024) remove redundant layers based on layer importance, while CoMe (Wang et al., 2025) and FFN Fusion (Bercovich et al., 2025) combine layer information to mitigate the degradation caused by direct removal. Since inter-layer pruning eliminates not only projection, attention, and FFN computation but also activation read/write, residual updates, normalization, and KV-cache access, it can directly improve practical inference latency. However, removing an entire coarse-grained structure induces a large activation distribution shift, and the resulting distortion can accumulate through subsequent layers, leading to severe quality degradation.

Intra-layer Pruning. Intra-layer pruning reduces the effective hidden dimension d by removing channels or hidden dimensions inside each layer. SlimGPT (Ling et al., 2024), SOBP (Wei et al., 2024), LoRAP (Li et al., 2024), and Olica (He & Lin, 2025) perform channel pruning using importance metrics based on OBS (Hassibi et al., 1993) or Wanda (Sun et al., 2023), while DISP-LLM (Gao et al., 2024) alleviates inter-layer structural dependency by changing hidden dimensions. When the hidden dimension is reduced from d to d', the overall computation becomes $O(L(nd'^2 + n^2d'))$. However, since all layers are still executed, block-level costs such as layer-wise scheduling, activation movement, residual connections, and normalization remain, so theoretical FLOP reduction may not translate into proportional end-to-end latency reduction.

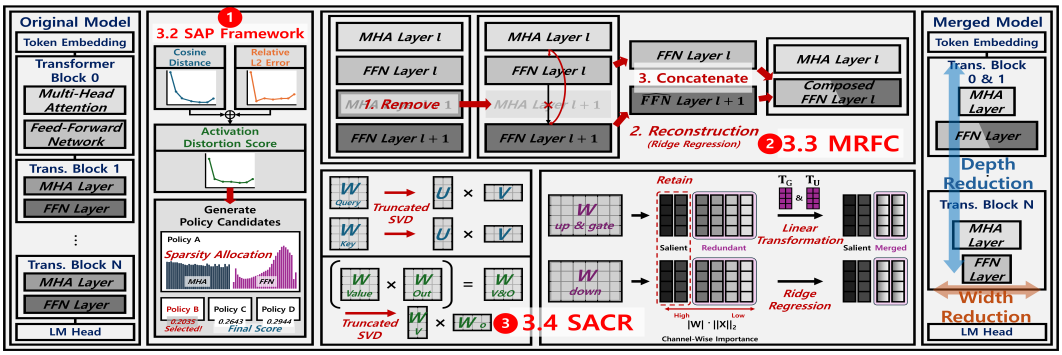


Figure 1: Overview of *PRISM*: activation-distortion-aware policy and inter-/intra-layer merging.

Hybrid-granularity Pruning. Hybrid-granularity pruning has been proposed to combine the strengths of inter-layer and intra-layer pruning while compensating for their individual limitations. Minitron (Sreenivas et al., 2024), Sheared-LLaMA (Xia et al., 2024) and LEMON (Chen et al., 2024) jointly reduce multiple structural units, including layers, heads, and intermediate dimensions, to improve the trade-off between quality and efficiency. However, selecting the pruning granularity and sparsity of each layer without recovery training remains challenging. These methods typically rely on post-pruning training, such as fine-tuning or knowledge distillation, to recover compression-induced performance degradation.

Overall, existing structured pruning methods reduce computation at different granularities, but they permanently remove the weights of structural units, causing loss of representation capability and requiring computationally expensive fine-tuning for recovery (***Lim. 1***). This motivates a method that adaptively selects pruning granularity and sparsity by considering layer-wise sensitivity and redundancy, without additional fine-tuning (***Lim. 2&3***).

3 Proposed Method: PRISM

3.1 Overall Architecture

PRISM is a policy-driven inter-/intra-layer merging framework that reformulates weight-removal-based structured compression as structured merging. Instead of discarding pruned structures, *PRISM* transfers their information into retained or merged structures. As shown in Figure 1, *PRISM* consists of three stages: (1) the SAP framework, which computes cosine distance and relative L2 error, combines them into ADS, and selects a sparsity allocation policy (***Lim. 3***), (2) MRFC, which removes selected MHA layers, reconstructs adjacent FFN representations via ridge regression, and concatenates neighboring FFNs for depth reduction (***Lim. 1&2***), and (3) SACR, which applies truncated SVD to MHA projections and reconstructs FFN channels by retaining salient channels and merging redundant ones for width reduction (***Lim. 1***). The full pipeline is provided in Algorithm 1 in the Appendix.

3.2 Activation-Distortion-Aware Sparsity Allocation Policy

PRISM introduces an ADS-based SAP to evaluate inter-/intra-layer merging operations under a unified criterion and determine inter-layer merging locations and layer-wise intra-layer sparsity. ADS quantifies how well an operation preserves the original activation by jointly considering directional and magnitude changes. As shown in Figure 2, cosine distance and relative L2 error capture only directional or magnitude distortion, respectively, and therefore do not reliably follow the layer-wise trend of PPL degradation under structured compression. In contrast, ADS combines both signals and better reflects the PPL degradation trend. *PRISM* therefore uses ADS as the criterion for sparsity allocation. Specifically, ADS combines cosine distance, obtained from cosine similarity, and normalized relative L2 error as follows:

$$\mathrm{ADS}(\mathbf{X},\widehat{\mathbf{X}}) = \alpha \cdot \underbrace{\mathbb{E}_{\mathcal{D}}\left[1 - \frac{\mathbf{X}^{\top}\widehat{\mathbf{X}}}{\|\mathbf{X}\|_2\|\widehat{\mathbf{X}}\|_2}\right]}_{\mathrm{CosDis}(\mathbf{X},\widehat{\mathbf{X}})} + \beta \cdot \underbrace{\mathbb{E}_{\mathcal{D}}\left[\frac{\|\mathbf{X}-\widehat{\mathbf{X}}\|_2}{\|\mathbf{X}\|_2}\right]}_{\mathrm{RelL2Err}(\mathbf{X},\widehat{\mathbf{X}})}. \tag{4}$$

Here, $\mathbf{X}, \widehat{\mathbf{X}} \in \mathbb{R}^{n \times d}$ denote the output activations of the original and merged models, respectively, and $\mathcal{D}$ is the calibration dataset. α and β are normalization factors that align the scales of the two metrics. For a given operation set, they are set to $1/\max(\text{CosDis})$ and $1/\max(\text{RelL2Err})$, respectively.

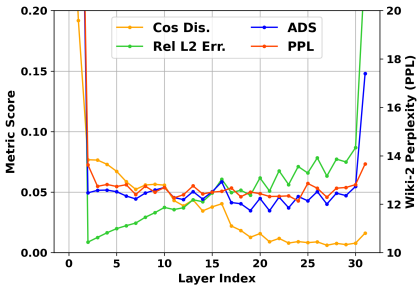


Figure 2: Layer-wise comparison of distortion metrics and PPL degradation under MHA layer removal on LLaMA2-7B.

SAP uses ADS as a distortion criterion to determine inter-layer merging positions and layer-wise intra-layer sparsity. It first evaluates four basic operations: Query–Key (QK) merging, Value–Output (VO) merging, FFN merging, and MHA layer removal. MHA layer removal is a binary operation and thus has one score per layer, whereas merging operations are evaluated at multiple sparsity budgets because their distortion depends on compression strength. Based on these scores, SAP constructs candidate operation combinations for the compressed model. In MHA, SAP assigns different compression ratios to QK and VO merging according to their relative distortion. For inter-layer merging, SAP removes MHA layers with low distortion scores and applies FFN layer composition to adjacent FFN layers. The composed FFN is then further compressed by intra-layer merging to reduce the increased intermediate-channel redundancy. Finally, SAP aggregates the scores of basic and composed operations, and selects the policy with the minimum total distortion among candidates satisfying the target compression level. The selected policy determines inter-layer merging locations, MHA/FFN sparsity allocation, and layer-wise intra-layer merging ratios.

3.3 MHA-Removed FFN Composition

Inter-layer merging reduces block-level execution paths and computation by merging coarse-grained layer structures. However, in Transformer blocks, MHA and FFN layers are alternately arranged; thus, directly merging two FFN layers separated by an MHA layer causes an input mismatch because they operate on different activations. To address this issue, *PRISM* proposes MRFC, which removes a selected MHA layer and composes the adjacent FFN layers into a single FFN. According to Eq. (1), the output of two blocks containing $\text{MHA}^{(l+1)}$ can be written as

$$\mathbf{X_A}^{(l+2)} = \text{FFN}^{(l+1)}\left(\mathbf{X_F}^{(l+1)}\right) + \text{MHA}^{(l+1)}\left(\mathbf{X_A}^{(l+1)}\right) + \text{FFN}^{(l)}\left(\mathbf{X_F}^{(l)}\right) + \mathbf{X_F}^{(l)}. \quad (5)$$

Here, $\text{FFN}^{(l)}$ and $\text{FFN}^{(l+1)}$ take $\mathbf{X_F}^{(l)}$ and $\mathbf{X_F}^{(l+1)}$ as inputs, respectively. Therefore, simple concatenation of the two FFN layers can induce activation distortion due to input-distribution mismatch. MRFC mitigates this issue through two steps: MHA layer removal and FFN layer composition.

MHA Layer Removal. In this step, *PRISM* measures the activation distortion caused by removing each MHA layer using ADS. Layers with low distortion are considered robust to removal and are used as inter-layer merging candidates. These scores are then used by SAP to determine the MRFC locations.

FFN Layer Composition. In this step, the two FFN layers adjacent after MHA layer removal are composed into a single FFN layer. Existing FFN Fusion (Bercovich et al., 2025) considers the fusion of consecutive FFN layers, but as shown in Eq. (5), the two FFN layers separated by an MHA layer operate on different input activations. *PRISM* therefore aligns the input activation of $\text{FFN}^{(l+1)}$ so that it can take $\mathbf{X_F}^{(l)}$ as input before composition. Specifically, it estimates a linear transformation $T_X \in \mathbb{R}^{d \times d}$ between $\mathbf{X_F}^{(l)}$ and $\mathbf{X_F}^{(l+1)}$ using the closed-form solution of ridge regression. The detailed formulation is provided in Appendix C.

$$\text{FFN}^{(l+1)}\left(\mathbf{X_F}^{(l+1)}\right) \approx \text{FFN}^{(l+1)}\left(\mathbf{X_F}^{(l)} T_X\right) = \widehat{\text{FFN}}^{(l+1)}\left(\mathbf{X_F}^{(l)}\right). \quad (6)$$

The estimated T_X is not retained as an additional operation; instead, it is absorbed into the gate and up projection weights of $\text{FFN}^{(l+1)}$. This converts $\text{FFN}^{(l+1)}$ into $\widehat{\text{FFN}}^{(l+1)}$, which takes $\mathbf{X_F}^{(l)}$ as input.

The down projection weights are then re-estimated by ridge regression to reduce the error between the transformed intermediate activation $\mathbf{Z}$ and the original output. Finally, $\mathrm{FFN}^{(l)}$ and $\widehat{\mathrm{FFN}}^{(l+1)}$ are concatenated to form a composed FFN layer:

$$\mathbf{X_A}^{(l+2)} \approx \mathrm{FFN}^{(l)}_{comp}\Big(\mathbf{X_F}^{(l)}\Big) + \mathbf{X_F}^{(l)}, \quad \mathrm{FFN}^{(l)}_{comp} = \mathrm{CONCAT}\Big\{\mathrm{FFN}^{(l)}, \widehat{\mathrm{FFN}}^{(l+1)}\Big\}. \quad (7)$$

Here, $\mathrm{FFN}^{(l)}_{comp}$ denotes the composed FFN layer. MRFC reduces block-level computation by removing the selected MHA layer. Since FFN layer composition increases the number of intermediate channels, the subsequent intra-layer merging stage further compresses these channels. Thus, MRFC is not simple layer removal, but an inter-layer merging operator that combines MHA layer removal with FFN layer composition to mitigate activation distortion. More detailed derivations are provided in Appendix D.

3.4 Structure-Aware Channel Reconstruction

While MRFC reduces layer-level redundancy, intra-layer merging addresses the remaining channel-level redundancy inside each layer. ***PRISM*** applies SACR, which reflects the structural properties of MHA and FFN layers.

MHA Layer Channel Reconstruction. For MHA layers, SACR performs truncated-SVD-based channel merging by exploiting low-rank redundancy in projection weights. Since Query and Key projections directly determine attention scores, $\mathbf{W}_q$ and $\mathbf{W}_k$ are compressed separately. In contrast, Value and Output projections can be viewed as consecutive linear transformations that map attention values back to the hidden dimension; therefore, SACR compresses the combined mapping of $\mathbf{W}_v$ and $\mathbf{W}_o$. This reduces MHA projection cost while preserving the structural correspondence of attention.

FFN Layer Channel Reconstruction. For FFN layers, SACR reconstructs redundant channels into a smaller number of merged channels rather than directly removing them, thereby reducing representation loss after the nonlinear interaction in the GLU (Shazeer, 2020) structure. SACR measures channel importance using both activation magnitude and down-projection weight contribution. The top-N important channels are kept as *salient* channels, while the bottom-R *redundant* channels are compressed into M *merged* channels ($M \ll R$). This avoids direct removal of low-importance channels and absorbs their representation into compact merged channels, mitigating activation distortion and representation loss.

The key step of SACR is to estimate linear transformations $T_G, T_U \in \mathbb{R}^{R\times M}$ that compress redundant GLU channels into merged channels. Let $\mathbf{G}_{red}, \mathbf{U}_{red} \in \mathbb{R}^{n\times R}$ denote the redundant-channel outputs of the gate and up projections, and let $\mathbf{G}_{mg}, \mathbf{U}_{mg} \in \mathbb{R}^{n\times M}$ denote the corresponding merged-channel outputs. The transformations are defined as

$$\mathbf{G}_{mg} = \mathbf{G}_{red}T_G, \quad \mathbf{U}_{mg} = \mathbf{U}_{red}T_U. \quad (8)$$

$$\mathbf{Z}_{mg} = \sigma(\mathbf{G}_{mg}) \odot \mathbf{U}_{mg} = \sigma(\mathbf{G}_{red}T_G) \odot (\mathbf{U}_{red}T_U) \approx \mathbf{Z}_{red}T_U. \quad (9)$$

Here, T_G and T_U are channel-compression matrices that map redundant gate and up channels to merged channels. Since GLU couples gate and up projections nonlinearly, directly removing redundant channels can introduce significant distortion in the intermediate activation. SACR estimates T_G and T_U from calibration activations so that the merged intermediate activation $\mathbf{Z}_{mg}$ preserves the contribution of redundant channels. The merged down-projection weight $\widehat{\mathbf{W}}_{d,mg}$ is then estimated by ridge regression to approximate the original output contribution of the redundant channels. The merged gate and up projection weights are obtained by multiplying the redundant weights by T_G and T_U, respectively. Since SACR is applied only to the redundant channel subset rather than the entire FFN, it can be used for both original FFN layers and MRFC-expanded composed FFN layers. Further derivations are provided in Appendix E.

Table 1: Performance comparison on LLaMA2-7B, LLaMA1-7B, and Vicuna-7B with dense baselines and SR 30%. The best score is marked in **bold**, and the second-best score is underlined.

Model	SR	Type	Method	PPL (↓)	Benchmark (↑)							Avg.
				Wiki-2	ARC-C	ARC-E	BoolQ	HellaS	OBQA	PIQA	WinoG	(%)
LLaMA2-7B	0%	–	Dense	12.19	46.33	74.58	77.71	76.00	44.20	79.05	68.98	66.69
	30%	Inter-layer	ShortGPT	126.46	33.70	45.33	62.20	50.91	31.60	63.55	62.98	50.04
			SLEB	27.64	31.23	51.39	48.99	53.72	33.00	68.88	52.88	48.58
			CoMe	30.70	35.32	53.54	54.77	61.73	37.00	71.55	61.32	53.60
		Intra-layer	LoRAP	19.36	34.81	58.67	67.16	66.46	38.40	74.76	65.90	58.02
			Olica	18.39	35.75	**61.70**	70.98	66.54	**39.60**	**75.30**	67.01	59.55
		Hybrid	**PRISM (Ours)**	**18.16**	**37.37**	61.36	**72.20**	**67.55**	39.20	**75.30**	**67.32**	**60.04**
LLaMA1-7B	0%	–	Dense	12.62	44.71	72.85	75.11	76.22	44.60	79.16	70.09	66.11
	30%	Inter-layer	ShortGPT	68.79	32.76	47.47	62.20	54.68	35.00	66.59	65.35	52.01
			SLEB	26.52	30.29	51.22	59.45	54.48	33.00	70.84	55.09	50.62
			CoMe	30.21	31.14	51.56	60.28	55.49	36.40	67.90	59.43	51.74
		Intra-layer	LoRAP	19.77	36.60	63.17	67.65	69.07	39.60	75.35	67.17	59.80
			Olica	18.85	37.63	62.92	68.44	68.84	**41.60**	75.14	67.17	60.25
		Hybrid	**PRISM (Ours)**	**18.71**	**38.82**	**64.73**	**71.74**	**69.41**	40.40	**75.52**	**67.96**	**61.23**
Vicuna-7B	0%	–	Dense	16.24	45.73	71.25	80.92	73.78	45.00	78.02	69.38	66.30
	30%	Inter-layer	ShortGPT	97.87	33.11	48.48	62.42	48.95	34.40	64.47	62.83	50.67
			SLEB	34.28	32.00	57.15	46.24	53.53	35.00	68.34	55.49	49.68
			CoMe	35.54	33.96	56.44	37.95	58.03	38.80	70.13	59.27	50.65
		Intra-layer	LoRAP	25.94	36.95	63.26	71.50	67.22	39.20	74.70	64.40	59.60
			Olica	23.76	37.97	**63.64**	74.22	67.53	**40.20**	**75.30**	**67.01**	**60.84**
		Hybrid	**PRISM (Ours)**	**22.74**	**38.48**	63.26	**75.20**	**67.67**	39.80	74.59	66.38	60.77

4 Experimental Results

4.1 Experimental Settings

Models, Datasets, Metrics and Baselines. We evaluate *PRISM* on open-source LLMs from the LLaMA-1 (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), and Vicuna (Chiang et al., 2023) families. For model quality, we use lm-evaluation-harness (Gao et al., 2021) and report zero-shot accuracy on seven downstream benchmarks: ARC-Challenge, ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), and BoolQ (Clark et al., 2019), together with their average accuracy. We also report perplexity (PPL) on WikiText-2 (Merity et al., 2016) with sequence length 128. As baselines, we compare with inter-layer structured pruning methods, including ShortGPT (Men et al., 2025), SLEB (Song et al., 2024), and CoMe (Wang et al., 2025), as well as intra-layer methods, including LoRAP (Li et al., 2024), and Olica (He & Lin, 2025). Our main comparisons are conducted under a retraining-free structured compression setting.

4.2 Main Results

Performance Comparison. Table 1 compares dense baselines and structured compression results at a sparsity ratio (SR) of 30% on LLaMA2-7B, LLaMA1-7B, and Vicuna-7B. *PRISM* achieves the lowest WikiText-2 PPL across all three models. On LLaMA2-7B, *PRISM* obtains a PPL of 18.16 and an average benchmark accuracy of 60.04, improving over the strongest intra-layer baseline, Olica, which achieves a PPL of 18.39 and an average accuracy of 59.55. On LLaMA1-7B, *PRISM* achieves a PPL of 18.71 and an average accuracy of 61.23, improving Olica by 0.14 PPL and 0.98 percentage points in average accuracy. On Vicuna-7B, *PRISM* also achieves the lowest PPL, reducing Olica's PPL from 23.76 to 22.74. Although its average accuracy is comparable to Olica (60.77 vs. 60.84), *PRISM* performs better on ARC-Challenge, BoolQ, and HellaSwag, showing that it improves PPL while largely preserving downstream performance on instruction-tuned models. Overall, inter-layer-only methods suffer from severe degradation at high sparsity due to large activation shifts, whereas intra-layer-only methods better preserve accuracy but have limited compression scope. These results

Table 2: Compression, latency, and PPL on LLaMA2-7B. Latency is measured for generating 256 tokens.

SR	Method	S_{mem}	S_{depth}	Latency (s)	PPL
0%	Dense	1.000	1.000	6.17 (× 1.00)	12.19
21%	CoMe	1.274	1.280	**4.78** (× 1.29)	24.45
	Olica	1.252	1.000	6.30 (× 0.98)	15.23
	PRISM (Ours)	1.260	1.032	5.84 (× 1.06)	**14.91**
30%	CoMe	1.438	1.455	**4.20** (× 1.47)	30.70
	Olica	1.409	1.000	5.71 (× 1.08)	18.39
	PRISM (Ours)	1.418	1.067	5.62 (× 1.10)	**18.16**
39%	CoMe	1.650	1.684	**3.91** (× 1.58)	53.86
	Olica	1.608	1.000	5.13 (× 1.20)	28.14
	PRISM (Ours)	1.619	1.143	4.90 (× 1.26)	**24.98**

Table 3: Wall-clock time (s) breakdown of each stage in the PRISM.

Process	21%	30%	39%
ADS Measurement	657.1	659.4	645.8
trunc-SVD (MHA)	129.2	128.1	124.6
SACR (FFN)	6.7	7.9	9.0
MRFC (Inter-layer)	27.8	47.6	67.5
SAP Selection	15.5	16.7	19.9
Other	30.8	24.6	19.7
Total	867.1	884.3	886.5

show that *PRISM* effectively mitigates PPL and downstream accuracy degradation across diverse 7B models by using ADS-based sparsity allocation to reflect layer-wise sensitivity. Additional results under various sparsity ratios are provided in Appendix F (Table 5).

4.3 Efficiency Analysis

Latency–PPL Trade-off. Table 2 compares compression and latency–PPL trade-offs on LLaMA2-7B across sparsity ratios. We report the memory-ratio reference $S_{\text{mem}} = M_0/M$ and depth-ratio reference $S_{\text{depth}} = L_0/L$, together with 256-token generation latency and WikiText-2 PPL, where M and L denote GPU memory usage and layer count, respectively, and subscript 0 denotes the dense model. At matched SRs of 21%, 30%, and 39%, *PRISM* consistently achieves lower latency and PPL than Olica. At SR 39%, their memory-ratio references are similar (1.619 for *PRISM* and 1.608 for Olica), while *PRISM* additionally reduces depth ($S_{\text{depth}} = 1.143$ versus 1.000). *PRISM* reduces latency from 5.13 s to 4.90 s and improves PPL from 28.14 to 24.98. CoMe achieves greater depth reduction ($S_{\text{depth}} = 1.684$) and the lowest latency of 3.91 s, but its PPL increases to 53.86. These results illustrate the benefit of combining inter- and intra-layer merging to improve latency while preserving quality at a matched compression budget. Across budgets, *PRISM* at SR 39% uses approximately 21% less GPU memory than CoMe at SR 21%, with a latency increase of 0.12 s and a PPL increase of 0.53. Detailed efficiency results and analysis are provided in Appendix F (Table 8).

Compression Pipeline Wall-clock Breakdown. Table 3 reports the wall-clock time of each stage in the *PRISM* compression pipeline on LLaMA2-7B. The total time is 867.1s, 884.3s, and 886.5s at SR 21%, 30%, and 39%, respectively, remaining stable as sparsity increases. Most of the time is spent on ADS measurement, which takes 645–659s across all sparsity ratios and dominates the pipeline cost. In contrast, the additional cost of truncated-SVD (MHA), SACR (FFN), MRFC (Inter-layer), and SAP selection remains limited. In particular, SACR takes only 6.7–9.0 s, while SAP selection requires 15.5–19.9 s. SAP keeps its overhead low by comparing candidate policies using precomputed ADS values, without repeatedly transforming the model. MRFC increases from 27.8s to 67.5s as SR increases, but still accounts for a small portion of the total pipeline time. These results show that the main cost of *PRISM* is concentrated in activation distortion measurement, while the additional policy-selection overhead remains modest.

4.4 Ablation Studies

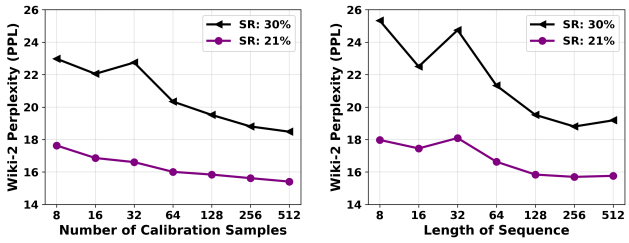
Effect of Distortion Metrics. Table 4 analyzes the effect of each distortion metric used in ADS-based sparsity allocation on LLaMA2-7B. Without the policy, fixed sparsity allocation yields PPLs of 15.87 and 18.97 at SR 21% and 30%, respectively. Relative L2 error alone sharply increases PPL, indicating that this metric alone cannot reliably guide allocation. Cosine similarity alone achieves PPLs of 15.43 and 18.79 but does not capture activation-scale errors. Combining both metrics in ADS yields the lowest PPLs of 14.91

Table 4: Ablation study on distortion metrics for ADS-based sparsity allocation.

Policy	RelL2Err	CosSim	SR 21%	SR 30%
			15.87	18.97
✓	✓		78.21	127.27
✓		✓	<u>15.43</u>	<u>18.79</u>
✓	✓	✓	**14.91**	**18.16**

and 18.16, supporting the joint use of directional and magnitude information for layer-wise sparsity allocation.

Effect of Calibration Samples. Figure 3 analyzes how calibration settings affect the PPL of *PRISM* on LLaMA2-7B. We randomly sample calibration data from C4 (Raffel et al., 2020) and vary the number of samples and sequence length. As shown in Figure 3, increasing the calibration scale generally reduces PPL. When the number of samples increases from 8 to 512, PPL decreases from 22.98 to 18.48 at SR 30% and from 17.63 to 15.54 at SR 21%. Similarly, when sequence length increases from 8 to 512, PPL decreases from 25.33 to 19.18 at SR 30% and from 17.97 to 15.76 at SR 21%. PPL fluctuations at 32 samples or tokens may reflect sampling variability. More calibration data supports reliable activation-distortion estimation and layer-wise sparsity allocation. We use 128 samples of 128 tokens to balance performance and calibration overhead.



(a) Number of samples (b) Sequence length

Figure 3: Effect of calibration sample count and sequence length on *PRISM* PPL under SR 21% and 30%.

Inter-/Intra-layer Trade-off. Figure 4 analyzes the trade-off between PPL and inference efficiency under different inter-/intra-layer merging ratios on LLaMA2-7B. The overall sparsity ratio is fixed to 30%, and we vary the ratio between the two merging granularities from inter-only to intra-only while measuring PPL and inference speedup. The x-axis denotes the inter-/intra-layer merging ratio, while the left and right y-axes show PPL and speedup, respectively. The inter-only setting provides large speedup by directly removing coarse-grained structures, but increases PPL to 43.71 due to large layer/block-level structural changes. In contrast, the intra-only setting reduces PPL to 18.05 by exploiting channel-level redundancy, but its speedup is limited to $1.06\times$ because it cannot sufficiently remove coarse-grained computation. Hybrid settings better balance quality and efficiency by exploiting both granularities. In particular, at an inter-/intra-layer ratio of 20:80, *PRISM* achieves a PPL of 18.16 and a $1.10\times$ speedup, providing additional acceleration while maintaining PPL close to the intra-only setting. These results indicate that jointly exploiting inter- and intra-layer redundancy allows *PRISM* to balance quality preservation and inference efficiency.

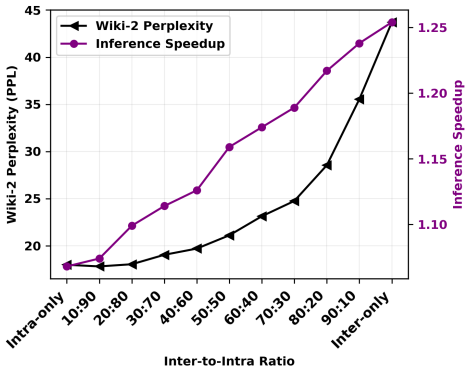


Figure 4: PPL and inference speed trade-off across inter-/intra-layer merging ratios under SR 30%.

5 Conclusion

We proposed *PRISM*, a retraining-free structured compression framework for LLMs. Instead of directly removing structural units, *PRISM* mitigates representation loss through structured weight merging, which transfers information from removed structures into retained structures. By combining MRFC, SACR, and ADS-based policy selection, *PRISM* jointly exploits inter- and intra-layer redundancy and performs sparsity allocation based on layer-wise sensitivity. Experiments on various 7B LLMs show that *PRISM* provides stable performance over existing structured pruning methods and achieves a balanced trade-off between model quality and inference efficiency. These results demonstrate that *PRISM* can improve practical inference efficiency while preserving LLM quality under a retraining-free setting.

AI USE STATEMENT

We used AI tools to assist with writing and language polishing, including revising sentence structure and improving the clarity, readability, and organization of the manuscript. For literature retrieval and discovery, AI tools helped formulate search queries and identify potentially relevant publications on structured LLM compression, inter- and intra-layer pruning, and model merging. In preparing the Background section, we used these tools to explore related terminology and research directions, identify candidate references, and locate prior approaches relevant to the scope of our method. These suggestions supported the collection of literature for further review. The authors are responsible for the final selection of cited works, the accuracy of bibliographic information and descriptions of prior work, and all claims and conclusions presented in the paper.

REFERENCES

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

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

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.

Akhiad Bercovich, Mohammed Dabbah, Omri Puny, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, Ehud Dov Karpas, Itay Levy, Zach Moshe, et al. Ffn fusion: Rethinking sequential computation in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Shiyao Cui, Tingwen Liu, Shuohuan Wang, Yu Sun, and Hua Wu. Lemon: Reviving stronger and smaller lms from larger lms with linear parameter fusion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8005–8019, 2024.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://lmsys.org/blog/2023-03-30-vicuna/ (accessed 14 April 2023)*, 2(3):6, 2023.

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.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL `https://arxiv.org/abs/2407.21783`.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Jiujun He and Huazhen Lin. Olica: Efficient structured pruning of large language models without retraining. In *International Conference on Machine Learning*, pp. 22580–22594. PMLR, 2025.

Young Chan Kim, Seok Kyu Yoon, Sung Soo Han, Chae Won Park, Jun Oh Park, Jun Ha Ko, and Hyun Kim. Accelerating language giants: A survey of optimization strategies for llm inference on hardware platforms. *Journal of Systems Architecture*, pp. 103690, 2026.

Gilhyeon Lee, Seonggeun Kim, Dongjun Lee, Kyungmin Goh, and Hyun Kim. Towards efficient language giants: A comprehensive survey on structural optimizations and compression techniques for large language models. *Neural Networks*, 201:108900, 2026.

Guangyan Li, Yongqiang Tang, and Wensheng Zhang. Lorap: Transformer sub-layers deserve differentiated structured compression for large language models. In *International Conference on Machine Learning*, pp. 28657–28672. PMLR, 2024.

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.

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.

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

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.

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

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.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *Proceedings of Machine Learning Research*, 235:46136–46155, 2024.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*, 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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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

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.

Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. Layer as puzzle pieces: Compressing large language models through layer concatenation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jiateng Wei, Quan Lu, Ning Jiang, Siqi Li, Jingyang Xiang, Jun Chen, and Yong Liu. Structured optimal brain pruning for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13991–14007, 2024.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. Finercut: Finer-grained interpretable layer pruning for large language models. In *Workshop on Machine Learning and Compression, NeurIPS 2024*, 2024.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A Appendix

B Experimental Setup

B.1 Models

LLaMA-1. LLaMA-1 (Touvron et al., 2023a) is an open-source autoregressive language model family. In our experiments, we use LLaMA-1 as a representative base LLM to evaluate whether PRISM can preserve language modeling and downstream task performance after structured compression.

LLaMA-2. LLaMA-2 (Touvron et al., 2023b) is an improved open-source language model family built upon the LLaMA architecture. We mainly use LLaMA-2 as the primary evaluation backbone, since it is widely adopted in recent LLM compression studies. Unless otherwise specified, our main experiments are conducted on LLaMA-2-7B. We additionally evaluate LLaMA-2-13B to examine the effectiveness of PRISM at a larger parameter scale within the same model family.

Vicuna. Vicuna (Chiang et al., 2023) is an instruction-tuned model derived from the LLaMA family. We include Vicuna to examine whether PRISM is also effective for instruction-tuned models, not only for base language models. This allows us to evaluate the applicability of PRISM across different model training stages.

LLaMA-3.1. LLaMA-3.1 (Grattafiori et al., 2024) is an autoregressive language model family that uses grouped-query attention (GQA). We evaluate LLaMA-3.1-8B to assess the applicability of PRISM to GQA-based models beyond the LLaMA-1 and LLaMA-2 families.

Qwen3. Qwen3 (Yang et al., 2025) is a multilingual language model family. We evaluate its dense Qwen3-8B and Qwen3-32B models, both of which use GQA, to examine the effectiveness of PRISM across model families and parameter scales.

B.2 Datasets

WikiText-2. WikiText-2 (Merity et al., 2016) is used to evaluate language modeling performance. We report perplexity (PPL) on WikiText-2, where lower PPL indicates better language modeling quality. Unless otherwise specified, PPL is evaluated with a sequence length of 128.

ARC-Challenge. ARC-Challenge (Clark et al., 2018) is a science question answering benchmark designed to evaluate challenging reasoning ability. We use zero-shot accuracy on ARC-Challenge to measure the model's reasoning capability after compression.

ARC-Easy. ARC-Easy (Clark et al., 2018) is the easier split of the ARC benchmark. It evaluates basic science question answering ability and is used together with ARC-Challenge to assess the robustness of compressed models on reasoning tasks.

HellaSwag. HellaSwag (Zellers et al., 2019) is a commonsense reasoning benchmark based on sentence completion. It evaluates whether the model can select the most plausible continuation among multiple candidates.

OpenBookQA. OpenBookQA (Mihaylov et al., 2018) is an open-book science question answering benchmark. It requires models to answer elementary-level science questions and is used to evaluate factual and reasoning capability.

PIQA. PIQA (Bisk et al., 2020) evaluates physical commonsense reasoning. The task requires selecting the more plausible solution to physical interaction scenarios, making it useful for measuring practical commonsense understanding.

WinoGrande. WinoGrande (Sakaguchi et al., 2021) is a commonsense reasoning benchmark based on pronoun resolution. It evaluates whether the model can correctly resolve ambiguous references using contextual and commonsense information.

BoolQ. BoolQ (Clark et al., 2019) is a yes/no question answering benchmark based on naturally occurring questions. We use BoolQ to evaluate the model's ability to answer binary questions from natural language context.

C4. C4 (Raffel et al., 2020) is a large-scale web text corpus constructed from Common Crawl. We use C4 as a calibration dataset for estimating activation statistics and compression scores. Since C4 contains diverse natural language documents, it provides broad calibration coverage for structured compression.

BookCorpus. BookCorpus (Zhu et al., 2015) is a large-scale collection of unpublished books. We use BookCorpus as an alternative calibration dataset to evaluate whether PRISM is robust to different calibration data distributions. Compared to C4, BookCorpus contains longer and more narrative-style text, allowing us to examine the effect of calibration domain on compression quality.

B.3 Baselines

ShortGPT. ShortGPT (Men et al., 2025) is an inter-layer structured pruning method that removes redundant Transformer layers using Block Influence (BI), which measures the change between layer input and output hidden states. We include it as a representative inter-layer pruning baseline.

SLEB. SLEB (Song et al., 2024) is an inter-layer structured pruning method that removes redundant Transformer blocks. It verifies block redundancy by comparing the model behavior before and after block removal, and eliminates blocks that have limited impact on the output. We include SLEB as a representative inter-layer pruning baseline.

CoMe. CoMe (Wang et al., 2025) is an inter-layer compression method that treats layers as reusable components and compresses LLMs through layer concatenation. It reduces model depth by combining information from multiple layers rather than simply removing them. We include CoMe as a strong inter-layer pruning baseline.

LoRAP. LoRAP (Li et al., 2024) is an intra-layer structured compression method that applies differentiated compression to Transformer sub-layers. It separately handles attention and FFN modules and reduces their internal dimensions using low-rank or structured approximation. We compare with LoRAP as a representative intra-layer compression baseline.

Olica. Olica (He & Lin, 2025) is a retraining-free intra-layer structured pruning method for LLMs. It compresses internal structures of Transformer blocks, such as attention and FFN components, without requiring post-pruning retraining. We include Olica as a strong retraining-free intra-layer baseline.

C Ridge Regression

Ridge regression augments the ordinary least-squares objective with an L_2 regularization term. Given a linear approximation $\mathbf{B} \approx \mathbf{AW}$, ridge regression estimates a stable transformation matrix by preventing the solution from overfitting to the calibration data. It is formulated as

$$\widehat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\mathbf{B} - \mathbf{AW}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 . \tag{10}$$

The closed-form solution is given by

$$\widehat{\mathbf{W}} = \left(\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I}\right)^{-1} \mathbf{A}^\top \mathbf{B}. \tag{11}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, $\mathbf{I}$ is the identity matrix, and λ is the regularization coefficient. Eq. (10) finds the transformation matrix $\widehat{\mathbf{W}}$ that minimizes the discrepancy between $\mathbf{B}$ and $\mathbf{AW}$, while Eq. (11) gives its closed-form solution. Using this formulation, we can estimate a linear transformation from $\mathbf{A}$ to $\mathbf{B}$ using only calibration data, without any retraining.

D DETAILS OF MRFC

Inter-layer compression commonly involves the question of which layers should be removed. Figure 5 compares the PPL changes caused by removing MHA and FFN layers. We observe that MHA layer removal induces relatively smaller performance degradation than FFN layer removal. Based on this observation, PRISM selects MHA layers as the removal targets in the inter-layer compression stage.

When an MHA layer is removed, the two FFN layers that were originally separated by the MHA layer become adjacent. FFN Fusion (Bercovich et al., 2025) shows that, when two SwiGLU-based FFN modules share the same input activation, the sum of their outputs can be equivalently represented as a single FFN by concatenating their intermediate channels. This property follows from the fact that an FFN output is composed of the sum of intermediate channel-wise contributions. Therefore, if two FFN layers share the same input $\mathbf{X_F}$, they can be represented as a single concatenated FFN.

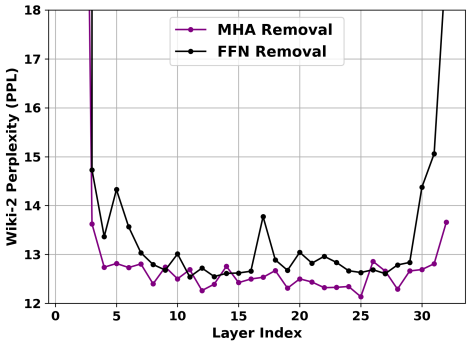


Figure 5: WikiText-2 PPL under MHA and FFN layer removal on LLaMA2-7B.

However, the two FFN layers merged in MRFC do not originally take the same input activation. Specifically, $\mathrm{FFN}^{(l)}$ takes ${\mathbf{X_F}}^{(l)}$ as input, whereas $\mathrm{FFN}^{(l+1)}$ takes ${\mathbf{X_F}}^{(l+1)}$, which is the activation after the removed MHA layer. Therefore, directly concatenating the two FFNs without any correction causes an input-distribution mismatch, leading to output errors in the concatenated FFN.

To mitigate this issue, MRFC first performs input alignment so that the later FFN layer can take the same input as the earlier FFN layer. Since MHA layer and residual connection operations exist between the input activations of two adjacent FFN layers, it is difficult to assume an exact linear relationship between them. Nevertheless, on calibration activations, the discrepancy between the two input distributions can be approximated by a linear transformation. Specifically, MRFC estimates a transformation matrix T_X using ridge regression such that ${\mathbf{X_F}}^{(l+1)} \approx {\mathbf{X_F}}^{(l)} T_X$. Then, instead of adding T_X as an explicit operation, MRFC absorbs it into the gate and up projection weights of $\mathrm{FFN}^{(l+1)}$:

$$\mathbf{G}^{(l+1)} = {\mathbf{X_F}}^{(l+1)} \mathbf{W}_g^{(l+1)} \approx {\mathbf{X_F}}^{(l)} T_X \mathbf{W}_g^{(l+1)} = {\mathbf{X_F}}^{(l)} \widehat{\mathbf{W}}_g^{(l+1)}, \tag{12}$$

$$\mathbf{U}^{(l+1)} = {\mathbf{X_F}}^{(l+1)} \mathbf{W}_u^{(l+1)} \approx {\mathbf{X_F}}^{(l)} T_X \mathbf{W}_u^{(l+1)} = {\mathbf{X_F}}^{(l)} \widehat{\mathbf{W}}_u^{(l+1)}. \tag{13}$$

Here, $\widehat{\mathbf{W}}_g^{(l+1)} = T_X \mathbf{W}_g^{(l+1)}$ and $\widehat{\mathbf{W}}_u^{(l+1)} = T_X \mathbf{W}_u^{(l+1)}$. By absorbing T_X into the projection weights, $\mathrm{FFN}^{(l+1)}$ is converted into a form that directly takes ${\mathbf{X_F}}^{(l)}$ as input, without introducing an additional linear layer.

However, even after aligning the gate and up projections, the intermediate activation $\mathbf{Z}$ is not perfectly preserved due to the SiLU nonlinearity and Hadamard product inside the FFN layer. Therefore, MRFC further reconstructs the down projection weight so that the transformed intermediate activation can reproduce the original output contribution of $\mathrm{FFN}^{(l+1)}$. Let the original output contribution be $\mathbf{D}^{(l+1)} = \mathbf{Z}^{(l+1)} \mathbf{W}_d^{(l+1)}$, and let $\widehat{\mathbf{Z}}^{(l+1)}$ denote the intermediate activation obtained from the input-aligned FFN. MRFC estimates $\widehat{\mathbf{W}}_d^{(l+1)}$ by solving the following ridge regression problem:

$$\widehat{\mathbf{W}}_d^{(l+1)} = \arg\min_{\mathbf{W}} \left\| \mathbf{D}^{(l+1)} - \widehat{\mathbf{Z}}^{(l+1)} \mathbf{W} \right\|_F^2 + \lambda \left\| \mathbf{W} \right\|_F^2. \tag{14}$$

This reconstruction reduces the remaining output-level approximation error after input alignment.

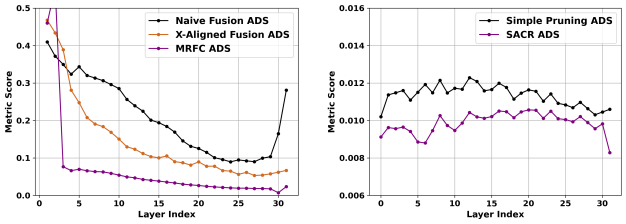
Finally, MRFC concatenates $\mathrm{FFN}^{(l)}$ and the input-aligned and down-reconstructed $\mathrm{FFN}^{(l+1)}$ to construct a composed FFN. Therefore, MRFC is not a simple MHA layer removal method. Instead, it is an inter-layer merging method that aligns the representation of the FFN layer around the removed MHA layer to the earlier FFN input and absorbs its output contribution into the retained computation path.

Figure 6a compares the effect of each MRFC component using ADS across layers. *Naive Fusion* denotes the case where only the MHA layer is removed without correcting the later FFN. *X-Aligned Fusion* applies only $\mathbf{X_F}$-alignment without down projection reconstruction. *MRFC* denotes the final method that applies both input alignment and down projection reconstruction. Although MRFC shows relatively large ADS in some early layers, after the second layer, Naive Fusion generally shows the highest ADS, X-Aligned Fusion achieves lower ADS, and MRFC achieves the lowest ADS. This indicates that input alignment with T_X reduces the input mismatch between FFN layers, while down projection reconstruction further mitigates the remaining output distortion. Therefore, the two correction steps in MRFC effectively reduce the representation loss caused by MHA layer removal.

E Details of SACR

As described in the main text, SACR reconstructs redundant FFN channels into a smaller number of merged channels instead of directly removing them. To this end, SACR uses two channel-compression linear transformations, T_G and T_U, for the redundant gate and up channels, respectively. For each merged channel, we first select a pivot channel from the redundant channel set. The pivot channel is selected based on the activation norm and represents the activation pattern that the corresponding merged channel should preserve. Then, each column of T_G is constructed by assigning the largest coefficient to the pivot channel and distributing smaller coefficients to the remaining redundant channels. This encourages the merged gate activation to follow the gate activation pattern of the pivot channel.

After constructing T_G, we determine T_U by considering the gate-side approximation error. Specifically, channels whose gate activations are similar to the pivot channel receive larger coefficients in T_U, while channels with larger gate mismatch receive smaller coefficients. This design reflects the fact that the FFN intermediate activation is computed as $\sigma(\mathbf{G}) \odot \mathbf{U}$, where errors in the gate projection can be amplified through the Hadamard product with the up projection. Therefore, T_U is not merely an up-channel merging matrix, but also serves to compensate for the gate approximation error. As a result, SACR approximates $\mathbf{Z}_{mg}$ to be close to $\mathbf{Z}_{red}T_U$, as described in Eq. (9).



(a) MRFC component ablation (b) SACR vs. simple pruning

Figure 6: Layer-wise ADS comparison of MRFC components and SACR.

The linear transformations T_G and T_U, defined at the activation level, are also reflected in the actual FFN weights. Let $\mathbf{W}_{g,red}$ and $\mathbf{W}_{u,red}$ denote the redundant gate and up projection weights, respectively. The merged gate and up projection weights are constructed as follows:

$$\mathbf{W}_{g,mg} = \mathbf{W}_{g,red}T_G, \qquad \mathbf{W}_{u,mg} = \mathbf{W}_{u,red}T_U. \tag{15}$$

Thus, T_G and T_U are absorbed into the existing gate and up projection weights rather than being introduced as additional runtime operations. Consequently, SACR reduces the FFN intermediate width without adding extra inference computation.

Finally, SACR reconstructs the down projection weight so that the merged intermediate activation can reproduce the original output contribution of the redundant channels. Let $\mathbf{Z}_{red}$ and $\mathbf{W}_{d,red}$ denote the intermediate activation and down projection weight of the redundant channels, respectively. The original output contribution of the redundant channels is then given by $\mathbf{Z}_{red}\mathbf{W}_{d,red}$. SACR obtains the merged down projection weight $\mathbf{W}_{d,mg}$ by solving the following ridge regression problem:

$$\mathbf{W}_{d,mg} = \arg\min_{\mathbf{W}} \|\mathbf{Z}_{red}\mathbf{W}_{d,red} - \mathbf{Z}_{mg}\mathbf{W}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 . \tag{16}$$

Table 5: Performance comparison on LLaMA2-7B, LLaMA1-7B, and Vicuna-7B with dense baselines and SR 21% and 39%.

Model	SR	Type	Method	PPL (↓)	Benchmark (↑)							Avg.
				Wiki-2	ARC-C	ARC-E	BoolQ	HellaS	OBQA	PIQA	WinoG	(%)
LLaMA2-7B	0%	–	Dense	12.19	46.33	74.58	77.71	76.00	44.20	79.05	68.98	66.69
	21%	Inter-layer	ShortGPT	43.89	36.26	55.89	62.17	62.67	37.20	70.40	66.22	55.83
			SLEB	18.65	34.30	59.26	61.50	62.53	35.60	74.32	58.64	55.16
			CoMe	24.45	41.21	62.50	73.33	69.14	**41.00**	73.94	68.43	61.36
		Intra-layer	LoRAP	15.52	38.82	67.05	72.32	71.09	39.80	76.88	67.88	61.98
			Olica	15.23	40.27	**67.59**	72.91	70.99	39.80	76.82	67.64	62.29
		Hybrid	**PRISM (Ours)**	**14.91**	**41.38**	**67.59**	**73.73**	**71.25**	40.40	**77.37**	**69.61**	**63.05**
	39%	Inter-layer	CoMe	53.86	26.37	37.54	40.58	45.44	30.40	63.49	53.12	42.42
		Intra-layer	Olica	28.14	**33.11**	**55.93**	66.24	60.44	**36.40**	71.06	58.33	54.50
		Hybrid	**PRISM (Ours)**	**24.98**	31.66	53.58	**69.39**	**60.62**	34.60	**71.22**	**64.56**	**55.09**
LLaMA1-7B	0%	–	Dense	12.62	44.71	72.85	75.11	76.22	44.60	79.16	70.09	66.11
	21%	Inter-layer	ShortGPT	30.53	36.43	58.46	62.17	62.01	38.40	71.06	67.17	56.53
			SLEB	20.36	34.47	58.54	68.23	62.37	37.00	73.61	62.12	56.62
			CoMe	20.92	38.82	61.74	72.69	68.28	40.20	74.37	68.11	60.60
		Intra-layer	LoRAP	16.08	**40.87**	66.79	**73.85**	72.59	**41.40**	**77.48**	67.56	62.93
			Olica	15.90	40.70	67.21	71.71	**72.86**	41.20	77.42	68.82	62.85
		Hybrid	**PRISM (Ours)**	**15.79**	**40.87**	**68.18**	73.43	72.41	**41.40**	76.88	**69.77**	**63.28**
Vicuna-7B	0%	–	Dense	16.24	45.73	71.25	80.92	73.78	45.00	78.02	69.38	66.30
	21%	Inter-layer	ShortGPT	47.11	38.65	59.05	64.50	62.40	37.40	68.28	66.22	56.64
			SLEB	22.65	34.04	62.37	59.69	61.52	37.60	73.78	58.96	55.42
			CoMe	28.19	40.27	62.42	40.46	66.92	40.80	71.82	**68.27**	55.85
		Intra-layer	LoRAP	20.76	40.96	66.79	75.96	70.52	41.00	**76.12**	67.17	62.65
			Olica	19.81	41.72	66.79	**76.97**	**70.98**	40.20	75.79	66.93	62.77
		Hybrid	**PRISM (Ours)**	**19.49**	**41.98**	**67.26**	76.57	70.42	**41.40**	75.90	67.09	**62.95**

The resulting $\mathbf{W}_{d,mg}$ is concatenated with the down projection weights corresponding to the preserved important channels to form the final compressed FFN layer. In this way, SACR absorbs the representation of redundant channels into merged channels, reducing the FFN width while preserving the output contribution of the original FFN.

Figure 6b compares the activation distortion of SACR with that of simple channel removal across layers. Simple removal completely discards the contribution of redundant channels, which can make the activation of the merged model deviate significantly from the original activation. In contrast, SACR transfers redundant representation from the gate and up projection into merged channels through T_G and T_U, and reconstructs the down projection using ridge regression. As a result, SACR achieves lower ADS than simple channel removal in most layers. This demonstrates that SACR effectively reduces activation distortion while decreasing the number of FFN channels, preserving redundant-channel representation within the merged channels.

F Additional Experimental Results

Performance across Sparsity Ratios. Table 5 presents the detailed performance comparison at SR 21% and 39%. At SR 21%, *PRISM* achieves the lowest PPL and the highest average benchmark accuracy among compressed models across LLaMA2-7B, LLaMA1-7B, and Vicuna-7B. Specifically, *PRISM* obtains a PPL of 14.91 and an average accuracy of 63.05% on LLaMA2-7B, a PPL of 15.79 and an average accuracy of 63.28% on LLaMA1-7B, and a PPL of 19.49 and an average accuracy of 62.95% on Vicuna-7B. At SR 39% on LLaMA2-7B, *PRISM* achieves a PPL of 24.98 and an average accuracy of 55.09%, compared with 28.14 and 54.50% for Olica and 53.86 and 42.42% for CoMe. Thus, *PRISM* also achieves the best PPL and average accuracy among the evaluated compressed models under more aggressive compression. These improvements are not merely due to the lower sparsity ratio. Rather, they indicate that *PRISM* effectively reflects layer-wise sensitivity through ADS-based sparsity allocation and jointly exploits inter-/intra-layer redundancy by combining MRFC

Table 6: Performance comparison on recent GQA-based models and larger models at approximately 30% sparsity. WikiText-2 PPL is evaluated with a sequence length of 128, and Avg. denotes the mean zero-shot accuracy over seven benchmarks. The best and second-best results among compressed models within each model group are marked in bold and underlined, respectively.

Model	SR	Type	Method	PPL (↓)	Benchmark (↑)							Avg.
				Wiki-2	ARC-C	ARC-E	BoolQ	HellaS	OBQA	PIQA	WinoG	(%)
LLaMA-3.1 8B	0%	–	Dense	14.32	53.50	81.27	82.02	78.86	44.60	81.18	74.27	70.81
	30%	Inter-layer	CoMe	75.43	26.79	40.32	49.82	39.54	32.60	65.02	52.80	43.84
		Intra-layer	Olica	44.64	34.64	53.03	**65.47**	54.44	32.80	**71.65**	63.38	53.63
		Hybrid	**PRISM (Ours)**	**39.91**	**38.74**	**59.01**	45.78	**61.72**	**36.80**	71.06	**68.19**	**54.47**
Qwen3-8B	0%	–	Dense	21.13	56.57	80.72	86.57	74.94	41.60	77.69	67.01	69.30
	30%	Inter-layer	CoMe	115.92	26.96	45.88	63.94	39.63	27.40	62.40	52.64	45.55
		Intra-layer	Olica	222.33	25.77	36.91	42.08	37.13	27.40	60.01	55.09	40.63
		Hybrid	**PRISM (Ours)**	**81.77**	**31.57**	**48.70**	**72.57**	**49.54**	**33.40**	**67.68**	**56.91**	**51.48**
LLaMA2-13B	0%	–	Dense	10.98	49.15	77.44	80.52	79.38	45.20	80.52	72.22	69.20
	30%	Inter-layer	CoMe	19.36	42.66	66.54	68.65	71.24	40.00	75.57	69.06	61.96
		Intra-layer	Olica	15.71	43.69	71.55	**76.39**	**74.76**	45.20	77.69	68.98	65.47
		Hybrid	**PRISM (Ours)**	**15.57**	**45.65**	**73.11**	72.17	74.18	**46.20**	**78.29**	**70.40**	**65.71**
Qwen3-32B	0%	–	Dense	16.84	61.01	83.29	86.51	82.61	46.60	82.05	72.85	73.56
	30%	Inter-layer	CoMe	47.28	36.43	55.30	64.07	59.63	37.00	69.26	58.96	54.38
		Intra-layer	Olica	38.04	48.89	67.72	80.46	74.62	39.20	74.81	**65.98**	64.53
		Hybrid	**PRISM (Ours)**	**32.34**	**49.83**	**71.59**	**82.17**	**74.92**	**41.80**	**77.64**	64.72	**66.10**

and SACR. In particular, MRFC mitigates representation loss caused by coarse-grained structural compression, while SACR reduces fine-grained degradation by merging channel-level redundancy within each layer. Therefore, *PRISM* effectively suppresses the PPL increase induced by structured compression and maintains competitive downstream performance not only at SR 30%, but also at SR 21% and 39%.

Performance on GQA and Larger Models. Grouped-query attention (GQA) (Ainslie et al., 2023) allows multiple query heads to share key and value heads while retaining separate query projections. With H query heads and H_{kv} key/value heads, each group contains H/H_{kv} query heads. For the same sequence length, head dimension, and numerical precision, native GQA requires H_{kv}/H of the KV-cache storage used by MHA. This sharing introduces dependencies across query heads, since modifying a shared key/value projection affects all query heads within its group.

Table 6 extends our evaluation to GQA-based model families and larger models, including LLaMA-3.1-8B, Qwen3-8B, LLaMA2-13B, and Qwen3-32B, at approximately 30% sparsity. We report WikiText-2 PPL with a sequence length of 128 and average zero-shot accuracy over seven benchmarks. Across all four models, *PRISM* achieves the lowest PPL and highest average accuracy among the evaluated compressed models.

For LLaMA-3.1-8B, *PRISM* and Olica use the same GQA-to-MHA unfolding transformation, with SRs audited relative to the native dense model. Under this setting, *PRISM* achieves a PPL of 39.91 and an average accuracy of 54.47%, compared with 44.64 and 53.63% for Olica and 75.43 and 43.84% for CoMe. *PRISM* leads on five of the seven benchmarks, while Olica achieves higher accuracy on BoolQ and PIQA. Thus, the improvement in average accuracy does not imply uniform gains across tasks.

On Qwen3-8B, *PRISM* achieves a PPL of 81.77 and an average accuracy of 51.48%, outperforming Olica (222.33 PPL and 40.63% accuracy) and CoMe (115.92 PPL and 45.55% accuracy). It leads on all seven benchmarks, with average accuracy gains of 10.85 and 5.93 percentage points over Olica and CoMe, respectively. However, a substantial gap remains relative to the dense model (21.13 PPL and 69.30% accuracy), indicating that quality preservation at this sparsity level remains challenging.

Table 7: Comparison with reimplemented Minitron variants on LLaMA-3.1-8B at approximately 4.5B parameters. We report zero-shot accuracy (%) on seven benchmarks. All methods use the same approximately 50M-token knowledge distillation (KD) protocol in the recovery setting. Bold indicates the best result within each setting.

Method	ARC-C	ARC-E	BoolQ	HellaS	OBQA	PIQA	WinoG	Avg.
Without KD								
Minitron-Width	22.78	30.35	37.83	29.36	27.80	53.05	51.07	36.03
Minitron-Depth	25.85	32.79	58.81	33.90	28.00	56.86	46.80	40.43
PRISM (Ours)	**28.41**	**44.23**	**70.18**	**46.74**	**31.00**	**62.73**	**60.69**	**49.14**
With approximately 50M-token KD								
Minitron-Width	27.13	44.57	47.83	49.60	31.80	66.54	**59.43**	46.70
Minitron-Depth	27.30	**47.69**	48.23	49.33	30.80	**68.34**	54.62	46.61
PRISM (Ours)	**27.99**	47.26	**60.67**	**49.96**	**32.00**	67.57	58.56	**49.15**

On LLaMA2-13B, *PRISM* achieves a PPL of 15.57 and an average accuracy of 65.71%. Its improvements over Olica (15.71 PPL and 65.47% accuracy) are modest, amounting to 0.14 lower PPL and 0.24 percentage points higher accuracy. The gains over CoMe are larger: PPL decreases from 19.36 to 15.57, and average accuracy increases from 61.96% to 65.71%. *PRISM* leads on five benchmarks, while Olica performs better on BoolQ and HellaSwag.

On Qwen3-32B, *PRISM* achieves a PPL of 32.34 and an average accuracy of 66.10%, compared with 38.04 and 64.53% for Olica and 47.28 and 54.38% for CoMe. Relative to Olica, this corresponds to a PPL reduction of 5.70 and an accuracy gain of 1.57 percentage points; the accuracy gain over CoMe is 11.72 percentage points. *PRISM* leads on six of the seven benchmarks, with Olica performing better on WinoGrande. These results extend the observed advantage in PPL and average accuracy to the 32B setting.

Comparison with Minitron. We compare *PRISM* with Minitron (Sreenivas et al., 2024) width- and depth-pruned variants reimplemented from the paper specifications on LLaMA-3.1-8B, matching all methods at approximately 4.5B parameters. We evaluate each compressed model both without recovery training and after the same abbreviated knowledge distillation (KD) protocol using approximately 50M tokens (16,384 tokens per step for 3,051 steps). As shown in Table 7, without KD, *PRISM* outperforms both Minitron variants on all seven benchmarks, achieving 49.14% average accuracy compared with 36.03% and 40.43% for the width and depth variants, respectively. Even after KD, the Minitron variants reach 46.70% and 46.61%, remaining below *PRISM* without recovery training. These results support the effectiveness of *PRISM* under constrained recovery budgets; they do not establish superiority over Minitron with its full recovery-training budget.

Details of Efficiency. Table 8 reports the detailed efficiency and latency–quality trade-offs on LLaMA2-7B. Latency is measured for generating 256 tokens, and language-modeling quality is evaluated using WikiText-2 PPL. We compare the measured speedup, $S_{\text{meas}} = T_0/T$, with three analytic references: $S_{\text{comp}} = C_0/C$, $S_{\text{mem}} = M_0/M$, and $S_{\text{depth}} = L_0/L$, where T, C, M and L denote latency, MACs, GPU memory usage and layer count, respectively, and subscript 0 denotes the dense model. MACs denote multiply–accumulate operations.

As sparsity increases, *PRISM* progressively reduces parameter count, MACs, GPU memory usage, and generation latency, accompanied by increasing PPL. At SR 39%, it reduces the model to 4.14B parameters, 494.98G MACs, and 7983 MiB of GPU memory, corresponding to reductions of 38.6%, 36.3%, and 38.3% relative to the dense model. Latency decreases from 6.17 s to 4.90 s, yielding a measured speedup of $1.259\times$, while PPL increases from 12.19 to 24.98.

At each evaluated SR, *PRISM* achieves both lower latency and lower PPL than Olica. At SR 21%, *PRISM* requires slightly more MACs than Olica (633.23G versus 625.59G), yet achieves lower latency (5.84 s versus 6.30 s) and better PPL (14.91 versus 15.23). At SR 39%, it reduces latency from 5.13 s to 4.90 s and improves PPL from 28.14 to 24.98. These comparisons show that MAC reductions alone do not determine practical generation latency under the evaluated setting.

Table 8: Efficiency and latency–quality trade-offs on LLaMA2-7B across sparsity ratios. Latency is measured for generating 256 tokens, and PPL is evaluated on WikiText-2. Measured speedup and analytic reference ratios are normalized to the dense model. The best and second-best PPL and measured speedup at each nonzero SR are marked in bold and underlined, respectively.

SR	Method	Model cost			Measured performance			Analytic references			$S_{\text{meas}}/S_{\text{mem}}$ (%)
		Params (B)	MACs (G)	Memory (MiB)	Latency (s)	S_{meas} (↑)	PPL (↓)	S_{comp}	S_{mem}	S_{depth}	
0%	Dense	6.74	777.32	12928	6.17	1.000	12.19	1.000	1.000	1.000	100.0
21%	CoMe	5.32	664.38	10150	4.78	**1.291**	24.45	1.170	1.274	1.280	101.3
	Olica	5.33	625.59	10324	6.30	0.979	<u>15.23</u>	1.243	1.252	1.000	78.2
	PRISM	5.33	633.23	10260	5.84	<u>1.057</u>	**14.91**	1.228	1.260	1.032	83.9
30%	CoMe	4.72	586.67	8992	4.20	**1.469**	30.70	1.325	1.438	1.455	102.2
	Olica	4.73	560.09	9173	5.71	1.081	<u>18.39</u>	1.388	1.409	1.000	76.7
	PRISM	4.73	566.83	9119	5.62	<u>1.098</u>	**18.16**	1.371	1.418	1.067	77.4
39%	CoMe	4.11	508.95	7834	3.91	**1.578**	53.86	1.527	1.650	1.684	95.6
	Olica	4.16	497.93	8040	5.13	1.203	<u>28.14</u>	1.561	1.608	1.000	74.8
	PRISM	4.14	494.98	7983	4.90	<u>1.259</u>	**24.98**	1.570	1.619	1.143	77.8

$S_{\text{meas}} = T_0/T$, $S_{\text{comp}} = C_0/C$, $S_{\text{mem}} = M_0/M$, and $S_{\text{depth}} = L_0/L$, where T, C, M, and L denote generation latency, MACs, GPU memory usage, and layer count, respectively; subscript 0 denotes the dense model. MACs denote multiply–accumulate operations.

CoMe achieves the lowest latency at every evaluated SR, accompanied by higher PPL than both *PRISM* and Olica. At SR 39%, CoMe reaches 3.91 s latency with a PPL of 53.86, whereas *PRISM* achieves 4.90 s with a PPL of 24.98. Thus, at similar parameter counts, *PRISM* preserves substantially better language-modeling quality while providing a smaller speedup.

The reference ratios help contextualize these measured speedups. In particular, S_{mem} is computed from GPU memory usage. For *PRISM*, $S_{\text{meas}}/S_{\text{mem}}$ is 83.9%, 77.4%, and 77.8% at SR 21%, 30%, and 39%, respectively, exceeding the corresponding Olica ratios of 78.2%, 76.7%, and 74.8%. These ratios describe the relationship between observed speedup and the memory-based reference, rather than hardware utilization. CoMe exceeds 100% at SR 21% and 30%, illustrating that this reference is not an upper bound on measured speedup.

Comparisons across different sparsity levels also require accounting for model cost. For example, CoMe at SR 21% has lower latency and PPL (4.78 s and 24.45) than *PRISM* at SR 39% (4.90 s and 24.98). However, the latter uses fewer parameters (4.14B versus 5.32B) and less GPU memory (7983 MiB versus 10150 MiB). The preferred configuration therefore depends jointly on the latency, quality, and memory requirements of the deployment setting.

G LIMITATIONS

Although our evaluation includes GQA-based model families and models up to 32B parameters, it remains limited to dense decoder-only LLMs, and extension to other architectures, such as mixture-of-experts models, requires further study. PRISM relies on calibration activations for policy selection and reconstruction, making it potentially sensitive to shifts in data domains and sequence lengths. Moreover, compression-induced quality degradation remains model-dependent and can be substantial, even when PRISM outperforms the compared baselines. Our detailed latency, memory, and compression-time measurements are restricted to LLaMA2-7B under the evaluated GPU setting; their generalization to larger models, longer generation workloads, and different inference runtimes requires further validation. Future work will investigate broader architectures, calibration robustness, and deployment settings.

H ALGORITHM

Algorithm 1 summarizes the ADS-based sparsity allocation and policy-driven merging process of PRISM. First, using the calibration data $\mathcal{D}$, PRISM collects the layer activations of the original model $\mathcal{M}$. It then temporarily applies each candidate operation at each layer and measures ADS between

the original activation and the modified activation. The candidate operations consist of MHA layer removal (MLR), MHA QK merging (QKM), MHA VO merging (VOM), and FFN merging (FNM). This allows PRISM to compare the activation distortion induced by different operations under a unified criterion. Here, ADS follows the definition in Eq. (4), where α and β are normalization factors used to align the scales of cosine distance and relative L2 error. Next, PRISM selects MRFC candidate locations using the distortion score ADS_{MLR} for MHA layer removal. A lower ADS_{MLR} indicates that removing the corresponding MHA layer causes a relatively small activation change, and thus the layer is regarded as a suitable location for MRFC. Since applying MRFC composes adjacent FFN layers into a single composed FFN and changes the intermediate-channel structure, the algorithm further introduces FFN-related candidate operations, FFN Composition (FNC) and composed FFN merging (CFM), for the MRFC-applied candidate models and re-measures their ADS. Based on the measured ADS values, PRISM constructs the candidate policy set Π and computes the total distortion score $\mathrm{Score}(\pi)$ for each policy. The final policy π^* is selected as the policy with the smallest total distortion among candidates that satisfy the target sparsity budget S. The selected policy π^* contains the inter-layer locations $\mathcal{I}_{\mathrm{MLR}}$ where MRFC is applied, as well as the layer-wise numbers of QK, VO, and FFN merged channels, denoted by $\{k_{\mathrm{QK}}^{(l)}\}$, $\{k_{\mathrm{VO}}^{(l)}\}$, and $\{k_{\mathrm{FFN}}^{(l)}\}$, respectively. Finally, PRISM applies MRFC according to π^*, performs truncated-SVD-based merging for MHA, and applies SACR for FFN merging to produce the final merged model $\widehat{\mathcal{M}}$.

Algorithm 1 ADS-based Sparsity Allocation Policy

Require: Model $\mathcal{M}$, calibration data $\mathcal{D}$, target sparsity budget S
Ensure: Merged Model $\widehat{\mathcal{M}}$

Step 1: Activation Distortion Measurement
Measure ADS for MHA layer removal, MHA QK merging, MHA VO merging, and FFN merging
Define candidate operations $\mathcal{O}^{(l)} = \{o_{\mathrm{MLR}}, o_{\mathrm{QKM}}, o_{\mathrm{VOM}}, o_{\mathrm{FNM}}\}$
$\mathcal{A} \leftarrow \textsc{ForwardCollect}(\mathcal{M}, \mathcal{D})$
for each layer l in $\mathcal{M}$ **do**
  for each candidate operation $o \in \mathcal{O}^{(l)}$ **do**
    $\widehat{\mathcal{M}}_{l,o} \leftarrow \textsc{ApplyOperation}(\mathcal{M}, l, o)$ ▷ Sec. 3.3–3.4
    $\widehat{\mathcal{A}}_{l,o} \leftarrow \textsc{ForwardCollect}(\widehat{\mathcal{M}}_{l,o}, \mathcal{D}, l)$
    $ADS_{l,o} \leftarrow \alpha \cdot \mathbb{E}_{\mathcal{D}}\left[1 - \frac{\mathcal{A}_l^\top \widehat{\mathcal{A}}_{l,o}}{\|\mathcal{A}_l\|_2 \|\widehat{\mathcal{A}}_{l,o}\|_2}\right] + \beta \cdot \mathbb{E}_{\mathcal{D}}\left[\frac{\|\mathcal{A}_l - \widehat{\mathcal{A}}_{l,o}\|_2}{\|\mathcal{A}_l\|_2}\right]$. ▷ Eq. (4)
  end for
end for
Step 2: Activation Distortion Re-Measurement ▷ Sec. 3.2–3.3
Select top-k MRFC locations with the lowest ADS_{MLR}
Apply MRFC to the selected candidate locations and construct candidate models
Define candidate operations $\mathcal{O}^{(l)} = \{o_{\mathrm{FNC}}, o_{\mathrm{CFM}}\}$
for each candidate MRFC-applied model $\mathcal{M}'$ **do**
  Re-measure ADS for FFN candidate operations on $\mathcal{M}'$
end for
Step 3: Construct candidate policies Π ▷ Sec. 3.2
for each candidate policy $\pi \in \Pi$ **do**
  Compute total distortion score $\mathrm{Score}(\pi)$
end for
Select the policy satisfying the target sparsity budget:
$\pi^* = \arg \min_{\pi \in \Pi,\ \mathrm{Cost}(\pi) \geq S} \mathrm{Score}(\pi)$
Step 4: Policy-driven Merging
Parse $\pi^* = \left(\mathcal{I}_{\mathrm{MLR}}, \{k_{\mathrm{QK}}^{(l)}\}, \{k_{\mathrm{VO}}^{(l)}\}, \{k_{\mathrm{FFN}}^{(l)}\}\right)$
for each layer l in $\mathcal{M}$ **do**
  if $l \in \mathcal{I}_{\mathrm{MLR}}$ **then**
    $\mathcal{M} \leftarrow \textsc{ApplyMRFC}(\mathcal{M}, l)$ ▷ Sec. 3.3
  else
    $\mathcal{M}_{\mathrm{MHA}}^{(l)} \leftarrow \textsc{MergeMHA}(\mathcal{M}_{\mathrm{MHA}}^{(l)}, k_{\mathrm{QK}}^{(l)}, k_{\mathrm{VO}}^{(l)})$ ▷ Sec. 3.4
  end if
  $\mathcal{M}_{\mathrm{FFN}}^{(l)} \leftarrow \textsc{ApplySACR}(\mathcal{M}_{\mathrm{FFN}}^{(l)}, k_{\mathrm{FFN}}^{(l)})$ ▷ Sec. 3.4
end for
$\widehat{\mathcal{M}} \leftarrow \mathcal{M}$
return $\widehat{\mathcal{M}}$

Notation. ForwardCollect collects layer activations on calibration data, and ApplyOperation temporarily applies a candidate operation for ADS measurement. ApplyMRFC, MergeMHA and ApplySACR denote the inter-layer, MHA, and FFN merging operations specified by π^*, respectively.