IntraSlice: Towards High-Performance Structural Pruning with Block-Intra PCA for LLMs

Anonymous ACL submission

Abstract

Large Language Models (LLMs) achieve strong performance across diverse tasks but face deployment challenges due to their massive size. Structured pruning offers acceleration benefits but leads to significant performance degradation. Recent PCA-based pruning methods have alleviated this issue by retaining key activation components, but are only applied between modules in order to fuse the transformation matrix, which introduces extra parameters and severely disrupts activation distributions due to residual connections. To address these issues, we propose IntraSlice, a framework that applies block-wise module-intra PCA compression pruning. By leveraging the structural characteristics of Transformer modules, we design an approximate PCA method whose transformation matrices can be fully fused into the model without additional parameters. We also introduce a PCA-based global pruning ratio estimator that further considers the distribution of compressed activations, building on conventional module importance. We validate our method on Llama2, Llama3, and Phi series across various language benchmarks. Experimental results demonstrate that our approach achieves superior compression performance compared to recent baselines at the same compression ratio or inference speed.

1 Introduction

Large Language Models have rapidly advanced in recent years (Abdin et al., 2024; Touvron et al., 2023a; Zhang et al., 2022), demonstrating exceptional performance across a wide range of tasks. However, their enormous parameter scale demands substantial computational resources, which has become a major bottleneck hindering their broader deployment. To address this, numerous model compression techniques (Zhu et al., 2024; Xu and McAuley, 2023) have been proposed to make LLMs more suitable for resource-constrained environments or edge devices.

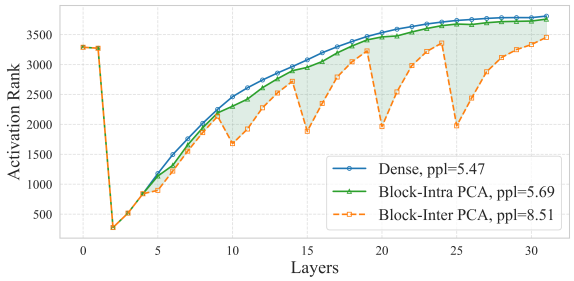

Figure 1: Comparison of rank changes in layer outputs under PCA compression within modules (Block-intra) vs. between modules (Block-inter) on LLaMA2-7B, applying 50% sparsity at layers 5, 10, 15, 20, and 25. Lines indicate output ranks, reflecting activation distributions. Block-inter PCA continuously affects the distribution of subsequent activations, keeping them in a low-rank state. In contrast, Block-intra PCA has minimal impact on activation distributions and achieves better results. More detail can be found in Appendix A.

Structured pruning removes entire components (attention heads or channels) based on model architecture, offering notable acceleration without requiring specialized frameworks or hardware. However, its major drawback is significant performance degradation, limiting practical use. Although recent methods (Ashkboos et al., 2024; Gao et al., 2024) have achieved promising results, performance loss remains a challenge.

Recently, PCA-based methods (Ashkboos et al., 2024) have been proposed to mitigate this issue by applying PCA to inter-module activations and retaining only the most important components during pruning. This effectively reduces compression error and alleviates pruning-induced performance degradation. However, **to fuse the transformation matrix into model weight, compression must be applied between modules,** they suffer from two major drawbacks: (1) The distribution shift in activations by pruning propagates through residual connections and accumulates across layers, as shown in Figure 1, resulting in severe distribution mismatch. (2) Because activation distributions vary

across modules, PCA matrices differ as well, requiring online computation in the residual path and significantly reducing overall acceleration.

To address these limitations, we propose IntraSlice, a PCA-based intra-module pruning method. In transformers-like architectures, (Liu et al., 2023) points out that the output amplitude of individual modules is often much smaller than that of the residual connections. Performing PCA-based compression within modules minimizes its impact on activation distributions and entirely eliminates the need for online computation in the residual connection. However, due to the presence of activation functions and nonlinear components in MHA and FFN, it is difficult to fuse the PCA transformation matrix directly into the weights. To resolve this, we combine PCA with the structural characteristics of the model and propose two novel techniques: Adaptive Head Compression with Block-PCA and Progressive Sliced Iterative PCA, which enable effective fusing of the PCA transformation matrix into weights while preserving its strong compression capability. Furthermore, we introduce a global pruning ratio evaluation method that simulates post-transformation activation distributions to allocate distinct pruning ratios across modules in a data-driven manner.

Our main contributions are:

- We propose a block-intra PCA solution for transformer modules. This effectively handles the nonlinearity of modules, achieving effective compression and allowing the transformation matrix to be fused into the model weights without introducing additional parameters.
- We introduce a global non-uniform pruning ratio evaluation method based on block-PCA, which considers conventional metrics and the distribution of compressed activations to provide more accurate pruning ratio estimation.
- We present a novel structured pruning framework for LLMs. Experimental results demonstrate that our method outperforms recent state-of-the-art approaches across a variety of benchmark tasks.

2 Related work

2.1 Semi-structured Pruning

Semi-structured pruning, especially N:M sparsity, is widely used in LLMs (Su et al., 2024; Meng et al., 2024), enforcing that at least N out of every M weights are pruned to enable efficient matrix operations, enabling efficient execution of matrix-multiply-accumulate operations. SparseGPT (Frantar and Alistarh, 2023) uses Optimal Brain Surgeon (OBS) to prune unimportant weights based on the Hessian and compensates for the induced perturbation. Wanda (Sun et al., 2024) and Plug-and-Play (Zhang et al., 2024b) further combine the input and output of weights to judge the importance of weights comprehensively. Prune-Zero (Dong et al., 2024) applies genetic algorithms to evolve importance metrics, treating pruning operations as genetic units. ProxSparse (Liu et al., 2025) adds sparse constraints and obtains the optimal pruning mask through overall optimization. SparseLLM (Bai et al., 2024) and LLM-surgeon (van der Ouderaa et al., 2024) extend OBS to global pruning via block-diagonal approximations or inter-block corrections, improving performance but at higher computational cost.

2.2 Structured Pruning

Compared to semi-structured pruning, structured pruning removes entire computational blocks (e.g., attention heads or channels), offering better hardware acceleration (An et al., 2024; Hu et al., 2024; Ashkboos et al., 2024; Gao et al., 2024). FLAP (An et al., 2024) assigns pruning ratios to different modules via a structured truncation metric. SliceGPT (Ashkboos et al., 2024) uses PCA on inter-module activations, preserving key components via transformation. SP^3 (Hu et al., 2024) applies PCA within heads and fine-tunes to recover performance. SoBP(Wei et al., 2024) leverages the gradient of each computational unit's mask as an importance score, combining with OBS for global optimization. DISP-LLM (Gao et al., 2024) relaxes the constraints imposed by conventional structural pruning methods and calculates the optimal width for the input/output dimensionalitys of each module. Moreover, Coarse-grained approaches like layer (Elhoushi et al., 2024; Kim et al., 2024) or module pruning (Zhang et al., 2024a; Wang et al., 2025a) offer greater speedups but with higher performance loss.

2.3 Global Non-Uniform Pruning

Global non-uniform pruning allocates different pruning ratios to modules based on their importance, aiming for optimal sparsity distribution (Yin et al., 2024; An et al., 2024). OWL (Yin et al.,

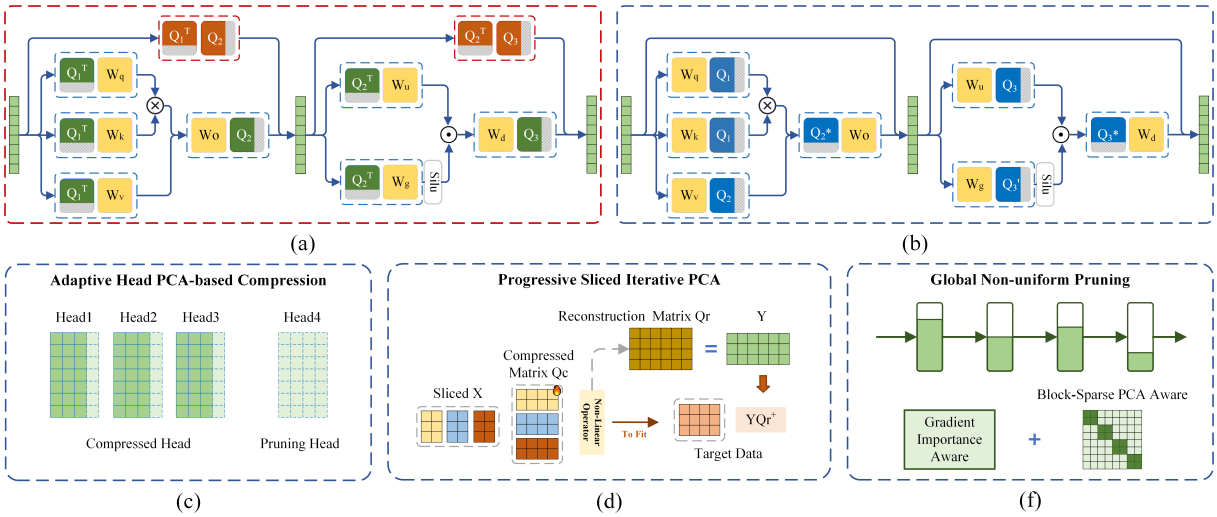

Figure 2: (a) The existing Inter-PCA pruning frameworks apply PCA compression between modules, which introduces additional computational overhead and error accumulation in residual paths. (b) Our IntraSlice framework (Intra-PCA) allows full fusion of matrices with less performance degradation. (c), (d) and (f) are the three components of IntraSlice, respectively.

2024) determines pruning ratios by evaluating each module's ability to reconstruct outlier parameters. BESA (Xu et al., 2024) employs mask learning to enable weight pruning with optimal, module-specific sparsity levels for large language models. SoBP(Wei et al., 2024) estimates the importance of each computational unit using the magnitude of its mask gradient and performs a global search to assign the most appropriate pruning ratio to each module. Týr-the-Prune (Li et al., 2025) constructs multiple pruning rate candidates for each layer and selects the optimal pruning rate candidate through iterative pruning and optimal search. Although recent methods improve pruning effectiveness, defining reliable evaluation metrics remains challenging.

3 Method

For an operation $Y = f(X)$, where X and Y are $\mathbb{R}^{N\times D}$. D and N denote data dimensionality and number, respectively. Q_c and Q_r are the corresponding compression matrices, where Q_c is $\mathbb{R}^{D\times P}$, and Q_r is $\mathbb{R}^{P\times D}$. P denotes the target compression dimensionality. The optimization of PCA between X and Y can be defined as:

$$\min_{Q_c,Q_r} \quad \|Y - f(XQ_c)Q_r\|_F^2 \\ s.t. \quad \begin{cases} Q_c^T Q_c = \mathbf{I} \\ Q_r Q_r^T = \mathbf{I} \end{cases} \tag{1}$$

When f is a linear operator, the problem reduces to a standard PCA formulation. However, the transformer architecture involve complex and diverse nonlinear structures. Specifically, MHA can only fuse block-diagonal matrices, while FFN, due to its stronger nonlinearity, cannot even fuse diagonal matrices. Therefore, we adopt structure-aware strategies tailored to different modules to maximize the reconstruction benefit of PCA. In practice, to improve reconstruction under complex nonlinearities, we removed orthogonality constraints and only limited the compression matrix's amplitude range.

The framework of our method is shown in Figure 2, which contains three main components. (1) **Adaptive head PCA-based compression**: The compression rate of each head is adaptively adjusted according to the structural characteristics of MHA, while taking into account both speed and accuracy. (2) **Progressive slicing iterative PCA**: *An optional operation* used to solve the nonlinear PCA optimization problem in FFN. (3) **Global non-uniform pruning**: Provide different pruning ratios for each block by jointly considering module importance and compressed activation distribution.

3.1 Adaptive Head PCA-based Compression

Structured head pruning removes entire attention heads, but this can be suboptimal—pruned heads may still carry useful information, while retained heads may be redundant. Ideally, assigning different compression ratios per head would improve retention of important features, but this breaks attention parallelism and adds time overhead. To address this, we propose an adaptive structured pruning strategy. IntraSlice removes only completely uninformative heads (pruned heads), while uniformly applying PCA compression to the rest

(compressed heads). This balances efficiency and performance without sacrificing parallelism.

3.1.1 Head reconstruction score.

Adaptive head compression relies on accurately estimating each head's importance and its reconstruction ability under different compression ratios. Following Wanda (Sun et al., 2024), we combine activations and weights to define a channel importance score I_i (Eq. 2), and obtain head importance R_h by summing its channel scores. R_h^p represents the reconstruction score when head h is pruned and compressed to p. According to PCA theory, the proportion of retained information corresponds to the proportion of the first p largest principal components. As shown in Eq. 3, R_h^p is computed based on the eigenvalues V of the Hessian matrix ($X^T X$, X is the output of a head) for head h, sorted in descending order. Each eigenvalue reflects the amount of information captured by its corresponding principal component.

$$R_h = \sum_{i \in h} I_i \quad ; \quad I_i = \|X_{:,i}\|_F^2 \cdot \|W_{i,:}\|_F^2 \tag{2}$$

$$R_h^p = R_h \cdot \frac{sum(V_{:p})}{sum(V)} \tag{3}$$

3.1.2 Greedy removal of heads.

Following PCA principles, we use reconstruction score maximization as the target. Heads with the lowest scores are greedily removed, and score gains of the remaining heads are evaluated. Let κ denote the current set of retained compressed heads, $|\kappa|$ be the number of such heads. P denotes the target compression dimensionality of the MHA block, defined as $P = (1 - r)D$, where D is the original feature dimensionality and r is the pruning ratio of the MHA block. The p represents the target dimensionality allocated to each remaining compression head, calculated as $p = P/|\kappa|$. When a head h_r is pruned, the available compression dimensionality p for the remaining heads increases to p^*, as shown in Eq. 5. The score gain S_g is defined as the improvement in the overall reconstruction score resulting from the removal of a head, as shown in Eq. 4. If the score gain S_g after removing a candidate head h_r is greater than zero, it is considered beneficial to remove h_r. Otherwise, h_r is retained as a compressed head.

This greedy iterative process efficiently approximates the optimal reconstruction score. Heads with negligible contribution are pruned, while the rest are compressed via PCA to retain key components.

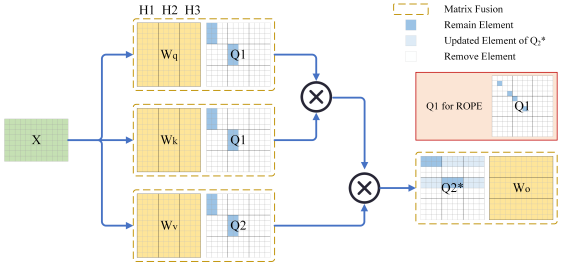

Figure 3: Schematic diagram of adaptive head PCA-based compression structure pruning and weight fusion.

This avoids the substantial information loss caused by blindly removing entire heads.

$$S_g = \sum_{h \in \kappa, h \neq h_r} R_h^{p^*} - \sum_{h \in \kappa} R_h^p \tag{4}$$

$$p^* = p + \frac{p}{|\kappa| - 1} = \frac{P}{|\kappa| - 1} \tag{5}$$

3.1.3 Structured pruning and weight fusion.

The attention module involves two PCA compression points: between value and output projections, and between query and key for attention score computation. However, the attention structure is inherently nonlinear, which complicates the integration of PCA transformation matrices. To address this, each head is treated as an independent compression unit as illustrated in Figure 3. The PCA transformation matrix Q_1 calculated by query and key is fused directly into the W_q and W_k, respectively. Specifically, to be fused into the value projection weight W_v, Q_2 is required to be a block-wise matrix. Importantly, it allows the transformation matrix Q_2^T fused into the W_o to be fully dense, meaning that the heads can compensate for others, as in Eq. 6, where X is the input of W_o.

$$Q_2^* = ((XQ_2)^T(XQ_2) + \lambda \mathbf{I})^{-1}(XQ_2)^T X \tag{6}$$

In models with Rotary Position Embedding (RoPE), the added nonlinearity complicates attention score computation. To enable efficient fusion, Q_1 is simplified into a pairwise channel selection matrix. Experiments show this has little impact on compression performance. More details of matrix fusion can be found in Appendix C.1.

3.2 Progressive Sliced Iterative PCA

Due to nonlinear activations and complex FFN structures, compression matrices in MLPs can't be directly fused into model weights. For such nonlinear compression problems, traditional iterative PCA requires loading full activation data repeatedly (Jolliffe, 2011), which is costly—e.g., in

LLaMA2-7B, extracting enough components may need thousands of iterations and heavy CPU-GPU transfers. Furthermore, constructing principal components using the entire activation dimensionality often fails to yield effective results due to the high complexity of the data. To address this issue, we propose a PCA method based on data slicing and progressive iteration (denoted as **IterPCA**). Unlike traditional PCA that splits along the compression dimension, our method slices by data dimension and incrementally updates the transformation matrix, reducing full data loading and lowering access and computation costs.

3.2.1 Iterative optimization with data slices.

As shown in Figure 2 (d), to further accelerate the process, we reformulate the optimization objective: instead of optimizing the reconstruction matrix Q_r, let $Y^r = YQ_r^+$ be the target, where Q_r^+ is the pseudo inverse of Q_r, Y^r is $\mathbb{R}^{N\times P}$. Here, Q_r can be obtained via Y as a suboptimal reconstruction matrix. For the optimization of Q_c, we draw inspiration from parallel acceleration in matrix computation, where a large dot-product operation can be decomposed into a sum of dot-products over smaller data blocks. We divide X into several d-dimensional parts, where d is the slice dimensionality of D after partitioning. In the k^{th} data slice, the corresponding optimization of the k^{th} part of Q_c is :

$$\min_{Q_c} \quad \left\| Y^r - f(X_{[:,(k-1)d:kd]}Q_{c[(k-1)d:kd]} + C) \right\|_F^2 \tag{7}$$
$$C = \sum_{i=1}^{k-1} X_{[:,(i-1)d:id]}Q_{c[(i-1)d:id]}$$

$$Q_r = (f(XQ_c)^T f(XQ_c) + \lambda \mathrm{I})^{-1} f(XQ_c)^T Y \tag{8}$$

Where C is the cumulative sum of the previous $k-1$ steps and has the same size as Y^r. Once Q_c is optimized, we recompute Q_r with Eq. 8. To achieve better results, we iteratively apply Eq. 7 and Eq. 8, updating Q_c and Q_r in sequence. Q_c is initialized as a channel selection matrix based on amplitude, while Q_r is initialized from Q_c using Eq. 8. In practice, iterative process is *an optional step* applied only when the current layer has a relatively high pruning ratio (around 10% of the layers). Mostly, we directly use the initialization results of Q_c and Q_r, so its overall time cost remains low.

3.3 Global Non-uniform Pruning

Global non-uniform pruning aims to assign different pruning ratios to blocks by evaluating the importance of computational units across layers. In SoBP(Wei et al., 2024), a mask is applied to each unit, and its gradient with respect to the final loss is used as an importance metric. To enhance accuracy, modern strategies often incorporate data compensation techniques such as OBS (Hassibi et al., 1993), which can alter the activation distribution used during pruning-rate estimation. Therefore, it is essential to consider compensation effects when estimating pruning ratios. To this end, we propose a PCA-based global non-uniform pruning-rate estimation method that accounts for both unit importance and the impact of PCA transformations.

3.3.1 Mask-based importance evaluation.

Following the importance calculation strategy introduced in SoBP(Wei et al., 2024), we apply a mask to each computational unit. The mask application scheme in MHA and FFN modules is illustrated in Eq. 9. $X_h^{l,i}$ is the output of the i^{th} head in l layer. Cat is the concatenation operation, M_h^l is the mask of MHA for l layer. Instead of setting one mask for the entire head, our method enables more fine-grained pruning within MHA module, setting a mask for each channel, M_h^l is $\mathbb{R}^D$, D is the hidden size. The mask of the FFN module is set similarly, M_f^l is $\mathbb{R}^{D_{inter}}$, D_{inter} is the intermediate size.

$$\begin{aligned} X_{mha}^l &= (Cat(X_h^{l,1}, X_h^{l,2}, ..., X_h^{l,H}) \circ M_h^l)W_o^l \\ X_{ffn}^l &= ((X_{mha}^l W_u^l) \circ \sigma(X_{mha}^l W_g^l)) \circ M_f^l)W_d^l \end{aligned} \tag{9}$$

According to the Taylor expansion, for layer l, the change in the final loss $\mathcal{L}$ for the mask m_i^l can be approximated as $g_i^l(m_i^l - 1)$, where g_i^l is the gradient of m_i^l when the mask is 1, m_i^l is the i^{th} mask of M_h^l or M_f^l. Then the importance I_i^l of this computational unit can be approximated with Eq. 10, where $\mathcal{D}$ is calibration data.

$$\begin{aligned} I_i^l &= (\mathcal{L}(m_i^l, \mathcal{D}) - \mathcal{L}(1, \mathcal{D}))^2 \approx (g_i^l)^2 \\ m_i^l &= 0 \end{aligned} \tag{10}$$

3.3.2 Sparse PCA-aware importance correction.

Since our method uses approximate PCA, we adopt sparse PCA to better approximate the transformation and accelerate computation. For layer l, let Q_s^l denote the transformation matrix derived from the block-PCA of activations X^l. Q_s^l is a block diagonal matrix ($\mathcal{R}^{D\times D}$ for MHA and $\mathcal{R}^{D_{inter}\times D_{inter}}$ for FFN). After transformation, the activation becomes $X^l Q_s^l$, and the corresponding weight becomes ${Q_s^l}^T W^l$. According to the backpropagation

Sparsity	Method	Llama2-7B		Llama2-13B		Llama2-70B		Llama3-8B		Phi-3-Medium-4k	
		PPL↓	Avg↑	PPL↓	Avg↑	PPL↓	Avg↑	PPL↓	Avg↑	PPL↓	Avg↑
0%	Dense	5.47	66.69	4.88	69.24	3.32	73.61	6.13	70.00	4.29	74.75
20%	SliceGPT	6.84	54.25	6.06	56.78	4.46	69.60	10.93	48.10	6.19	66.43
	Wanda	7.38	61.25	6.66	61.00	4.1	70.86	122.41	37.53	6.82	69.52
	FLAP	7.16	56.62	6.31	61.55	4.12	71.60	–	–	–	–
	SoBP	6.53	63.27	5.62	67.73	3.88	71.24	8.74	63.56	6.27	72.29
	SVD-LLM	8.52	49.60	6.78	58.56	–	–	47.00	45.43	7.16	68.02
	IntraSlice	**6.27**	**63.73**	**5.48**	**67.94**	**3.85**	**72.95**	**8.27**	**65.21**	**5.80**	**73.06**
30%	SliceGPT	8.64	46.70	7.44	50.10	5.41	61.61	17.02	41.40	7.52	56.82
	Wanda	9.17	56.28	10.14	44.72	4.77	69.90	271.71	36.50	10.00	62.41
	FLAP	8.85	50.91	7.57	57.27	4.82	69.68	–	–	–	–
	SoBP	7.58	59.15	6.27	66.82	4.36	70.30	10.32	58.60	7.05	67.52
	SVD-LLM	10.95	45.14	8.21	52.16	–	–	101.56	40.80	8.22	61.62
	IntraSlice	**7.11**	**60.49**	**5.96**	**67.13**	**4.34**	**72.27**	**10.25**	**60.65**	**6.71**	**69.73**
40%	SliceGPT	12.80	41.47	10.60	44.31	7.08	52.00	30.80	37.39	10.19	45.25
	Wanda	14.33	43.05	21.34	41.35	5.82	66.36	4258.41	34.68	20.68	53.52
	FLAP	11.49	48.70	9.07	53.18	6.24	67.96	–	–	–	–
	SoBP	9.28	56.06	7.39	60.86	4.96	68.58	12.48	**52.71**	8.02	61.15
	SVD-LLM	16.58	39.37	11.26	44.61	–	–	207.99	36.71	10.76	52.51
	IntraSlice	**8.39**	**56.38**	**6.92**	**62.24**	**4.91**	**69.99**	**12.28**	51.10	**7.90**	**64.90**

Table 1: Comparison of model compression results of different methods. The bold ones indicate the best results. PPL is the result on wikitext2, and Avg is the average result of 7 zero-shot tasks.

chain rule, the gradient of the mask M_h^l or M_f^l is transformed as $g^l Q_s^l$, where g^l represents the original gradient of the corresponding mask of the l layer before PCA transformation. The corrected importance score of single mask m_i^l is then calculated based on this adjustment with Eq. 11.

$$I_i^l = ((g^l Q_s^l)_i)^2 \tag{11}$$

Due to structural differences, the importance of MHA and FFN is not directly comparable. We generally assign the same pruning rate to both, and search for the optimal pruning rate for MHA or FFN respectively, by maximizing the sum of the importance scores I of the retained computing units. To control pruning tendency, we introduce a bias λ_b, scaling the MLP pruning ratio to $\lambda_b \times prune\_ratio$, while adjusting the MHA pruning ratio to maintain overall sparsity. This offers a more intuitive balance than tuning per-component importance (Wei et al., 2024). In practice, setting $\lambda_b \approx 1$ is often sufficient (see Appendix G for details).

4 Experiments

4.1 Experiment Setup

4.1.1 Models.

We evaluate our IntraSlice on several LLMs with a transformer architecture. Specifically, we consider models from the LLaMA-series (Touvron et al., 2023b; Dubey et al., 2024): LLaMA2-7B, LLaMA2-13B, LLaMA2-70B, and LLaMA3-8B; Phi-series (Abdin et al., 2024; Javaheripi et al., 2023): Phi-3-Medium-4K.

4.1.2 Baselines & tasks.

We compare our method with several recent state-of-the-art structured pruning approaches, including Wanda (Sun et al., 2024), SliceGPT (Ashkboos et al., 2024), FLAP (An et al., 2024), SVD-LLM (Wang et al., 2025b)and SoBP (Wei et al., 2024). In addition, we also compared with some more expensive methods(consume more data or longer time for pruning), such as DISP-LLM (Gao et al., 2024), týr-the-Pruner (Li et al., 2025) and LLM-surgeon (van der Ouderaa et al., 2024). Following SoBP(Wei et al., 2024), we adopt the llm-eval-harness (Gao et al., 2021) to evaluate our method and other compared methods on 7 widely-used zero-shot tasks: ARC-c, ARC-e (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018) and PIQA (Bisk et al., 2020).

4.1.3 Implementation details.

All experiments are conducted with PyTorch using the Hugging Face Transformers library on a single NVIDIA A800 GPU, with an exception for the 70B model which requires two GPUs. Following the SoBP(Wei et al., 2024) and SliceGPT (Ashkboos et al., 2024), we use a calibration set of 128 samples from the WikiText2 (Merity et al., 2017) training set, with a sequence length of 2048. We also evaluate on the C4 dataset to further validate

Sparsity	Model	PPL	WinoGrande	PIQA	OBQA	HellaSwag	BoolQ	ARC_e	ARC_c	Avg
0%	Llama2-13b	4.88	72.3	80.52	45.2	79.38	80.58	77.53	49.23	69.25
50%	SoBP	9.22	**64.48**	53.92	35.40	**58.13**	69.66	28.41	25.43	47.92
	IntraSlice	**8.75**	63.30	**66.87**	**37.00**	56.78	**69.85**	**58.38**	**33.53**	**55.10**
60%	SoBP	15.62	**59.27**	56.53	28.40	45.60	**66.27**	33.04	26.96	45.15
	IntraSlice	**11.36**	57.70	**62.30**	**32.60**	**46.40**	61.71	**48.06**	**29.35**	**48.30**

Table 2: Comparison of model compression results with high sparsity.

the effectiveness of our approach (see Appendix D.1 for C4 results).

4.2 Main Pruning Results

The perplexity (PPL) and zero-shot task accuracies of compressed models using different methods are reported in Table 1. Our IntraSlice consistently outperforms other approaches in most experiments. Compared to SliceGPT (Ashkboos et al., 2024), an inter-PCA compression method, our intra-PCA-based method demonstrates clear advantages, particularly under high sparsity. It is worth noting that SliceGPT (Ashkboos et al., 2024) introduces additional parameters at each residual connection, which undermines its acceleration efficiency. In comparison to the latest state-of-the-art method, SoBP(Wei et al., 2024), our approach exhibits stable improvements across well-trained models such as LLaMA2-7B, LLaMA2-13B, and LLaMA3-8B. Only in the LLaMA3-8B at 40% sparsity does the zero-shot accuracy slightly decrease. For the larger model LLaMA2-70B, our IntraSlice method substantially improves zero-shot task performance, while achieving slightly better perplexity than the baseline. We further tested at higher sparsity levels, as shown in Table 2. When the sparsity reaches 50% and 60%, our IntraSlice shows a more significant improvement over SoBP.

Sparsity	Method	E-T	PPL	Avg
0%	Llama2-7B	–	5.12	68.99
30%	LLM-Surgeon	✗	7.83	59.03
	DISP-LLM	✓	6.85	58.10
	týr-the-Pruner	✗	7.00	57.92
	IntraSlice	✗	**6.61**	**62.13**
50%	LLM-Surgeon	✗	15.38	45.68
	DISP-LLM	✓	**9.84**	46.72
	týr-the-Pruner	✗	10.44	49.10
	IntraSlice	✗	10.11	**50.85**
0%	Llama2-13B	–	4.25	71.79
30%	LLM-Surgeon	✗	6.21	65.34
	DISP-LLM	✓	5.77	63.07
	týr-the-Pruner	✗	6.05	64.75
	IntraSlice	✗	**5.63**	**68.27**
50%	LLM-Surgeon	✗	9.43	55.24
	DISP-LLM	✓	**7.11**	54.50
	týr-the-Pruner	✗	9.96	53.49
	IntraSlice	✗	7.72	**58.86**

Table 3: Comparison results of our method with LLM-surgeon, týr-the-Prune and DISP-LLM methods. In order to align with DISP-LLM, we only tested 5 zero-shot tasks. The PPL is tested on wikitext2 with sequence length 4096. E-T means end-to-end training.

In Table 3, we present a detailed comparison between IntraSlice, DISP-LLM, týr-the-Pruner, and LLM-Surgeon on LLaMA2-7B and LLaMA2-13B. The proposed IntraSlice achieves consistent perplexity and zero-shot accuracy improvement over týr-the-Pruner and LLM-Surgeon. When compared with DISP-LLM, our IntraSlice achieves remarkably higher zero-shot accuracy for both models at all the sparsity levels. Additionally, while DISP-LLM performs well in reducing perplexity, their performance in zero-shot tasks is poor compared with LLM-Surgeon and our IntraSlice. We attribute this to its end-to-end training paradigm and more training data (2×10000 samples), which highlights the potential of overfitting associated with DISP-LLM. Additionally, under lower sparsity levels, our method outperforms DISP-LLM, even using only 128 calibration samples. At the same time, in zero-shot tasks, IntraSlice consistently outperforms DISP-LLM by a large margin at all sparsity levels, which shows the strong generalization ability of the proposed IntraSlice.

4.3 Speedup Test

The fusion mechanism of our approach is illustrated in Figure 3. For models based on the transformer architecture, our method can be fully integrated into the model weights without requiring additional computations. More details of matrix fusion can be found in Appendix C.

The acceleration performance of LLaMA2-13B using our method is shown in Figure 4. Our method is comparable to current state-of-the-art structured

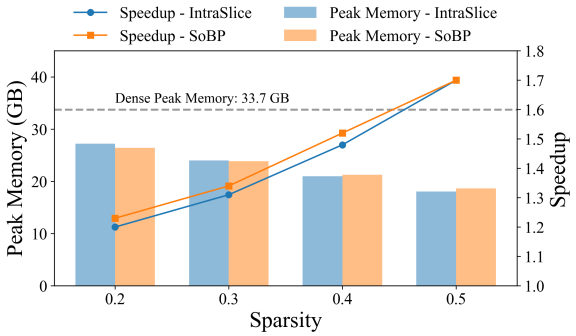

Figure 4: Inference speedup and memory of SoBP and IntraSlice at different sparsities, on the llama2-13B model.

Ada-H	Pca-A	IPca	20% Sparsity PPL↓	30% Sparsity PPL↓
✗	✗	✗	6.54	7.74
✓	✗	✗	6.32	7.28
✓	✓	✗	6.29	7.14
✓	✓	✓	**6.27**	**7.11**

Table 4: Impact of each component on Llama2-7B.

pruning methods in terms of acceleration and memory usage, but IntraSlice can achieve better compression performance. More details of speedup can be found in Appendix F.

4.4 Ablation Study

4.4.1 Impact of components.

We perform ablation studies to evaluate the impact of each component. Ada-H refers to the application of an adaptive PCA-based head pruning strategy. Pca-A uses a sparse PCA correction based on gradient evaluation when estimating the global pruning ratio, while IPca applies an iterative optimization process on the results of direct initialization within the FFN module. The naive baseline consists of global non-uniform pruning based on only mask gradients, un-iterated initialization of Q_c and Q_r in FFN, and traditional entire head pruning in MHA. As shown in Table 4, Ada-H significantly improves PPL performance. While Pca-A and IPca achieve relatively small improvements compared to Ada-H, they both consistently improve compression performance. Notably, Pca-A enhances sensitivity to important heads, reducing the effect of redundancy during pruning evaluation.

4.4.2 Impact of pruning bias λ_b.

We introduce a pruning bias λ_b to control the pruning tendency. As shown in Figure 5 (b), we tested the impact of the pruning bias on PPL and zero-shot accuracy. When λ_b is 1.0, that is completely balanced, it is comparable to the best result. The λ_b settings used in our paper can be found in Appendix D.2.

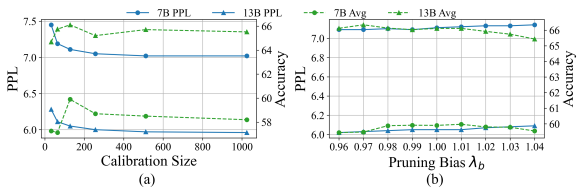

Figure 5: The impact of the size of calibration data and pruning bias λ_b on PPL and zero-shot accuracy on Llama2-7B and Llama2-13B with 30% sparsity.

4.4.3 Impact of the number of calibrations.

As shown in Figure 5 (a), we evaluate the effect of the number of calibration data on the results. As the number of calibration data increases, PPL decreases steadily until it starts to slow down around 1024. However, the accuracy of zero-shot tasks does not consistently improve with the increase in the amount of calibration data.

5 Conclusion

In this work, we propose a novel structured pruning framework for large language models that leverages block-level intra-module PCA to alleviate the performance degradation commonly seen in traditional pruning methods. To address the integration challenges of intra-PCA within MHA and FFN components, IntraSlice introduces an adaptive, head-wise PCA-based compression strategy along with a progressive, sliced iterative PCA. This design facilitates the seamless fusion of intra-PCA transformation matrices into model weights while retaining strong compression effectiveness. To better estimate the global non-uniform pruning ratio, we apply sparse PCA to capture post-compression activation patterns, enabling more accurate estimation of the global non-uniform pruning ratio. Extensive experiments demonstrate that IntraSlice achieves state-of-the-art results across multiple models, delivering significant improvements in pruning performance compared to existing methods. We believe this PCA-driven framework is a key step toward high-performance model compression. In our future work, we will aim to develop more flexible and efficient intra-PCA pruning strategies.

Limitations

While IntraSlice is effective, it has limitations. First, the iterative PCA introduces modest computational overhead when pruning, especially for large models, and its optimization remains suboptimal. Solving PCA under nonlinear structural constraints in transformers is still challenging. Second, for attention variants like Grouped Query Attention (GQA), query and key projections must be compressed consistently within each group to enable matrix fusion, imposing structural constraints that limit pruning flexibility. Addressing these challenges without sacrificing performance is a promising research direction.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone, 2024. *URL https://arxiv. org/abs/2404.14219*.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. 2024. SparseLLM: Towards global pruning of pre-trained language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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)*, pages 2924–2936.

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

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. 2024. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *Forty-first International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. In *ACL (1)*.

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

Leo Gao, Jonathan Tow, Stella Biderman, Shawn Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jasmine Hsu, Kyle McDonell, Niklas Muennighoff, and 1 others. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9.

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

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

Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. 2024. Sp³: Enhancing structured pruning via pca projection. In *ACL (Findings)*, pages 3150–3170.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Ian Jolliffe. 2011. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened LLaMA: A simple depth pruning for large language models. In *ICLR 2024*

Workshop on Mathematical and Empirical Understanding of Foundation Models.

Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. 2025. Týr-the-pruner: Structural pruning LLMs via global sparsity distribution optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hongyi Liu, Rajarshi Saha, Zhen Jia, Youngsuk Park, Jiaji Huang, Shoham Sabach, Yu-Xiang Wang, and George Karypis. 2025. PROXSPARSE: REGULARIZED LEARNING OF SEMI-STRUCTURED SPARSITY MASKS FOR PRETRAINED LLMS. In *Forty-second International Conference on Machine Learning*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and 1 others. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. 2024. Alps: Improved optimization for highly sparse one-shot pruning for large language models. In *NeurIPS*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. 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*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Yupeng Su, Ziyi Guan, Xiaoqun Liu, Tianlai Jin, Dongkuan Wu, Graziano Chesi, Ngai Wong, and Hao Yu. 2024. Llm-barber: Block-aware rebuilder for sparsity mask in one-shot for large language models. *arXiv preprint arXiv:2408.10631*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. 2024. The LLM surgeon. In *The Twelfth International Conference on Learning Representations*.

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

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025b. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*.

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

Canwen Xu and Julian McAuley. 2023. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10566–10575.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. BESA: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *The Twelfth International Conference on Learning Representations*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, AJAY KUMAR JAISWAL, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. 2024. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *Forty-first International Conference on Machine Learning*.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji

Kawaguchi. 2024a. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*.

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

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Rank Computation of Activation

To evaluate the rank of activations in large language models, we adopt the widely used Energy-Based Rank Selection method with a threshold of 99%. As illustrated in Figure 1, the activation distribution in the early layers is largely dominated by the input embeddings. Due to their uniformly distributed channels, these embeddings exhibit a high rank. In the first few transformer layers, however, outlier channels begin to emerge and dominate the activations, leading to a sharp drop in rank. As computation proceeds, the number of informative outlier channels gradually increases, resulting in a slow recovery of the rank.

To integrate the compression matrix into the model weights, inter-module PCA methods such as SliceGPT require compressing both the residual path and the module output. This leads to significant distributional shifts that can propagate to downstream layers via residual connections. In contrast, our method, IntraSlice, only modifies the output of the current module. This minimizes disruption to the overall activation distribution and has a negligible impact on subsequent layers.

B Algorithm

Algorithm 1 presents an overview of the IntraSlice framework. Algorithm 2 details the adaptive head compression procedure for the MHA module, while Algorithm 3 outlines the progressive iterative PCA compression applied to the FFN module. For clarity and concreteness, we illustrate the implementation using the LLaMA architecture as an example.

C Analysis of MHA matrix fusion

During pruning of the MHA module, IntraSlice eliminates low-information heads based on PCA analysis and applies uniform compression to the remaining heads to achieve effective compression. Since this process alters the dimensionality of attention heads, different fusion strategies are required depending on the specific MHA architecture. Nonetheless, all variants support full integration of the compressed structure.

C.1 Query and Key Compression with ROPE

RoPE applies distinct rotational position encodings to tokens at different positions. Conventional compression transformation matrices cannot pass through the RoPE operation and thus cannot be fused into the model weights. To enable matrix fusion, we construct a pairwise channel selection matrix adapted to the structure of RoPE. By leveraging its pairwise encoding properties, we assess channel importance and prune unimportant channels in pairs, as illustrated in Figure 6. We tested RoPE sparsity's effect on attention output in Llama2-7b. As shown in Figure 7. At 25% sparsity, average output relative error is 7.43%; at 50%, 27.8%. When the sparsity is within 25% (Keep Rate), the sparsity of the rope has a small impact on the output. We set minimum dimension of "compressed head" to 96 (original 128) to preserve performance.

In models with ROPE, where positional encoding is tied to specific channels, a minimal binary mask is introduced to track pruned query and key channels, which is negligible in practice.

Algorithm 1 Prune Algorithm of IntraSlice

Input: Sparsity r, Calibration $\mathcal{D}$, Model
Parameter: Pruning bias λ_b
Output: Model after pruning

Part 1: Global non-uniform prune ratio evaluation
$G, X \leftarrow CrossEntropyLoss(\text{Model}, \mathcal{D})$
$Q_S \leftarrow SparsePCA(X)$
$I \leftarrow (GQ_s)^2$
$S^H = \{S_1^H, ..., S_L^H\}, S^F = \{S_1^F, ..., S_L^F\}$
$\leftarrow GetGlobalPruningRatio(I, \lambda_b, r)$
for $l = 1$ **to** L **do**
  Part 2: Adaptive head PCA-based compression
  $mha \leftarrow l^{th}$ MHA module
  $X_l^H \leftarrow$ inputs of l^{th} MHA module
  $mha \leftarrow Algorithm2(S_l^H, mha, X_l^H)$
  Part 3: Progressive sliced iterative PCA
  $ffn \leftarrow l^{th}$ FFN module
  $X_l^F \leftarrow$ inputs of l^{th} FFN module
  $ffn \leftarrow Algorithm3(S_l^F, ffn, X_l^F)$
end for
return solution

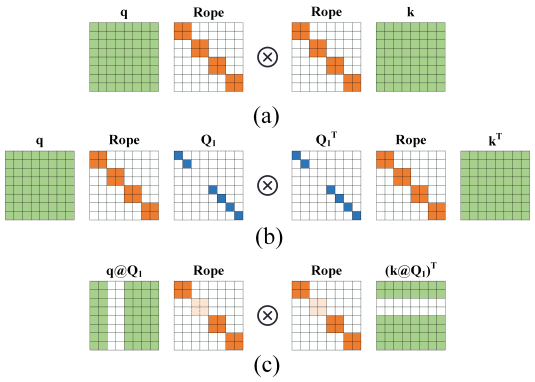

Figure 6: Schematic diagram of query and key compression with ROPE. Q_1 is the pairwise selection matrix. (a) Calculation of $q \times k$ with ROPE; (b) Construction of pairwise selection matrix Q_1; (c) Fusion Q_1 into query and key weights.

Algorithm 2 Adaptive Head PCA-based Compression

Input: Sparsity r, Inputs X, MHA

$q\_proj, k\_proj, v\_proj, o\_proj \leftarrow$ MHA
 # Compression of v and o
$X_o \leftarrow$ inputs of o_proj
$V \leftarrow$ PCA of X_o
$W_o \leftarrow$ weights of o_proj
$W_v \leftarrow$ weights of v_proj
$R \leftarrow HeadReconstructionScore(X_o, W_o)$ ▷ Eq.1
$remove\_head \leftarrow \varnothing$
for $\_ = 1$ **to** H **do**
  $h_r \leftarrow$ head with lowest score R
  $S_g \leftarrow ReconstructionScoreGain(h_r, V, R, r)$ ▷ Eq.3,4
  if $S_g > 0$ **then**
    $remove\_head \leftarrow remove\_head \cup \{h_r\}$
  end if
end for
$p \leftarrow$ head dim after compression
$Q_2 \leftarrow \text{PCA}(X_o, p)$
$Q_2^* \leftarrow Correct(Q_2^T, X_o)$ ▷ Eq.5
$W_o \leftarrow FusionMatrix(W_o, Q_2^*)$
$W_v \leftarrow FusionMatrix(Q_2, W_v)$
 # Compression of q and k
$Out_q, Out_k \leftarrow$ outputs of q_proj and k_proj
$Q_1 \leftarrow \text{PCA}(Out_q, Out_k, p, remove\_head)$
$W_q \leftarrow$ weights of q_proj
$W_k \leftarrow$ weights of k_proj
$W_q \leftarrow FusionMatrix(Q_1, W_q)$
$W_k \leftarrow FusionMatrix(Q_1, W_k)$
return MHA

Algorithm 3 Progressive Sliced Iterative PCA

Input: Sparsity r, Inputs X, FFN

$up\_proj, gate\_proj, down\_proj \leftarrow$ FFN
$W_u \leftarrow$ weights of up_proj
$W_g \leftarrow$ weights of $gate\_proj$
$W_d \leftarrow$ weights of $down\_proj$
$X_d \leftarrow$ inputs of $down\_proj$
$Out_u, Out_g \leftarrow$ outputs of up_proj and $gate\_proj$
Initialize Q_c, Q_r
$Q_c \leftarrow SelectMatrix(X_d, W_d, r)$
$Q_r \leftarrow OptimizeQ_r(Q_c, X_d)$ ▷ Eq.8
if Need Iteration **then**
  for $i = 1$ **to** $iteration\_number$ **do**
    $Q_c \leftarrow SliceOptimizeQ_c(Q_c, Q_r, Out_u, Out_g, X_d)$ ▷ Eq.7
    $Q_r \leftarrow OptimizeQ_r(Q_c, X_d)$ ▷ Eq.8
  end for
end if
$W_u \leftarrow FusionMatrix(Q_c, W_u)$
$W_g \leftarrow FusionMatrix(Q_c, W_g)$
$W_d \leftarrow FusionMatrix(W_d, Q_r)$
return FFN

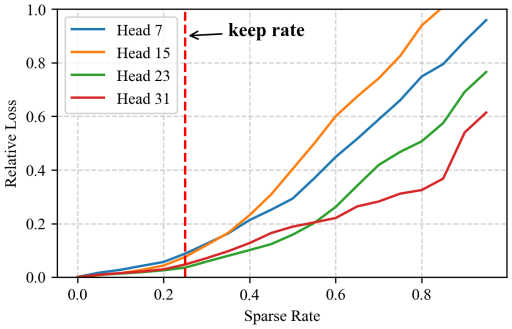

Figure 7: On Llama2-7b, 5th layer, the relative loss of attention output for different heads under different sparsity in ROPE with pairwise channel selection matrix.

C.2 Matrix Fusion of Llama Family

For the MHA of the ROPE llama architecture, the fusion of its transformation matrix is shown in Figure 8.

C.3 Matrix Fusion of Llama Family with GQA

As shown in Figure 9, for MHA using GQA, the query and key Q_1 of a head within a group are the same, marked with a red rectangle. The construction of Q_2^* is unaffected; only group-wide analysis is required when constructing Q_2.

D More Ablation Study

D.1 Ablation Study of C4 Dataset

To further verify the effectiveness of our method, we tested it on the C4 dataset and compared it with the current SOTA method SoBP. To maintain consistency in the settings, we use a calibration set of 128 samples from C4 train set with a length of 2048. PPL was tested on Wikitext2 with a length of 2048.

As shown in Table 5, even when evaluated on the C4 dataset, IntraSlice consistently achieves better perplexity (PPL) compared to SoBP. Across both LLaMA2-7B and LLaMA2-13B, our method significantly outperforms SoBP in terms of both PPL and zero-shot accuracy. These results highlight the effectiveness and strong generalization capability of IntraSlice.

D.2 Setup of Prune Bias λ_b

When constructing the global non-uniform pruning ratio, we introduce a pruning bias to adjust the relative sparsity between the MHA and FFN modules. While setting this bias to 1 yields good results for most models and sparsity levels, certain models benefit from a more tailored setting to achieve optimal performance. Table 6 summarizes the bias configurations used for different models and sparsity levels in this work.

Sparsity	Model	PPL	WinoGrande	PIQA	OBQA	hellaswag	BoolQ	ARC_e	ARC_c	Avg
			acc	acc-norm	acc-norm	acc-norm	acc	acc-norm	acc-norm	
0%	Llama2-7b	5.47	68.98	79.05	44.2	76.02	77.74	74.58	46.25	66.69
20%	SoBP	9.36	64.48	76.22	37.4	68.99	70.7	68.56	41.13	61.07
	IntraSlice	**7.11**	**68.43**	**76.88**	**40.2**	**72.68**	**75.66**	**69.02**	**41.98**	**63.55**
30%	SoBP	12.82	62.35	73.5	34.6	65.04	70.52	61.45	38.57	58.00
	IntraSlice	**8.87**	**67.64**	**75.08**	**38.4**	**68.84**	**75.2**	**65.03**	**38.05**	**61.18**
0%	Llama2-13b	4.88	72.3	80.52	45.2	79.38	80.58	77.53	49.23	69.25
20%	SoBP	8.5	70.56	77.91	42.8	77.23	80.73	74.41	46.25	67.13
	IntraSlice	**5.86**	**72.38**	**79.49**	**44.0**	**77.49**	**81.96**	**74.92**	**47.27**	**68.21**
30%	SoBP	11.0	69.69	76.93	39.6	73.07	80.73	70.62	43.77	64.92
	IntraSlice	**6.99**	**71.11**	**77.64**	**41.8**	**75.35**	**81.31**	**71.38**	**45.31**	**66.27**

Table 5: Comparison of model compression results of different methods with the C4 calibration dataset.

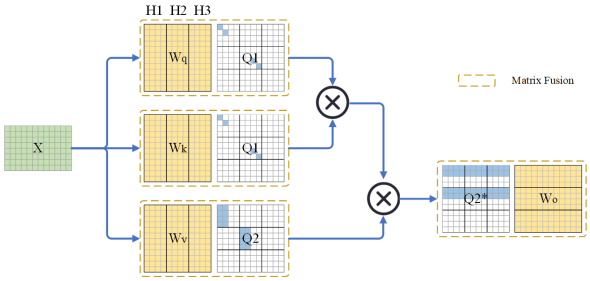

Figure 8: Schematic diagram of fusion Q_1, Q_2 and Q_2^* into llama MHA weights with ROPE.

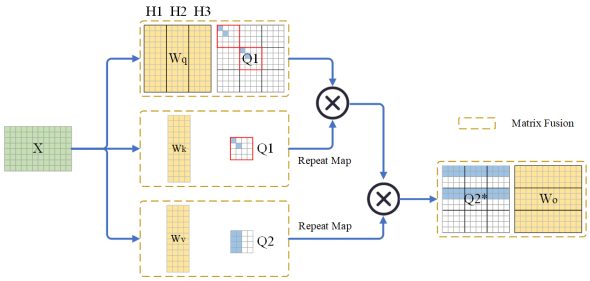

Figure 9: Schematic diagram of fusion Q_1, Q_2 and Q_2^* into llama MHA weights with ROPE and GQA.

E Compression Time

As shown in Table 7, IntraSlice achieves comparable compression times compared to recent state-of-the-art methods. The primary computational bottleneck lies in processing the intermediate dimensionality of the MLP layers. Additionally, block-wise PCA for global non-uniform pruning also contributes significantly to the overall time cost, which only needs to be computed once for all sparsity levels. In practice, all models except LLaMA2-70B can be pruned on a single 80 GB A800 GPU. In comparison, DISP-LLM and LLM-Surgeon require 4 and 8 such GPUs for pruning LLaMA2-7B and 13B, respectively.

Model	20%	30%	40%
LLaMA2-7B	1.00	1.00	1.00
LLaMA2-13B	1.00	0.82	1.00
LLaMA2-70B	0.82	0.91	0.94
LLaMA3-8B	0.98	0.95	1.00
Phi-3-medium-4k	1.00	1.00	1.00

Table 6: Setup of λ_b of different model and different sparsity.

Method	Llama2-7B	Llama2-13B	Llama2-70B
SliceGPT	0.20h	0.23h	1.97h
SoBP	0.46h	0.82h	8.54h
IntraSlice	0.52h	0.97h	7.38h
týr-the-Prune	13.06h	27.93h	–
DISP-LLM	4.82h	35.32h	–
LLM-Surgeon	68.52h	267.44h	–

Table 7: Comparison of compression time (GPU hours) of different methods on various models. SliceGPT, SoBP and IntraSlice are executed on the A800, while DISP-LLM and LLM-Surgeon are executed on the A100 and H100, respectively.

F Speedup Test

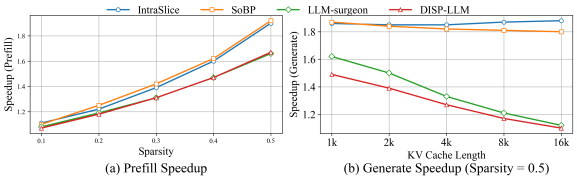

Figure 10: Speedup of different compression methods at different sparsities, on the llama2-13B model.

The acceleration performance of LLaMA2-13B using our method is shown in Figure 10. To accurately measure the speedup of different methods, we tested a single decoder layer to remove the effects of embedding and random sampling. Sparsity is the average of all layers. For all speed experiments, the prefill speed is evaluated with a

Sparsity	Method	PPL	WinoGrande	PIQA	HellaSwag	ARC-e	ARC-c	Avg
			acc	acc-norm	acc-norm	acc-norm	acc-norm	
0%	Llama2-7B	5.12	69.14	79.11	75.99	74.58	46.15	68.99
30%	LLM-Surgeon	7.83	61.09	73.56	60.72	63.09	36.69	59.03
	DISP-LLM	6.85	62.27	71.82	63.43	59.81	33.19	58.10
	týr-the-Pruner	7.00	64.17	74.27	66.41	52.74	32.00	57.92
	IntraSlice	**6.61**	**66.93**	**74.59**	**68.16**	**63.59**	**32.23**	**62.13**
50%	LLM-Surgeon	15.38	52.57	64.36	40.29	44.91	26.28	45.68
	DISP-LLM	**9.84**	54.54	63.93	46.33	43.06	25.85	46.72
	týr-the-Pruner	10.44	55.64	65.89	50.83	45.37	27.73	49.10
	IntraSlice	10.11	**61.09**	**64.8**	**50.21**	**48.36**	**29.78**	**50.85**
0%	Llama2-13B	4.25	80.52	72.3	79.38	77.53	49.23	71.79
30%	LLM-Surgeon	6.21	68.67	76.5	71.52	69.74	40.27	65.34
	DISP-LLM	5.77	74.43	66.85	70.86	63.8	39.42	63.07
	týr-the-Pruner	6.05	69.06	76.93	72.66	64.48	40.61	64.75
	IntraSlice	**5.63**	**76.71**	**71.98**	**73.87**	**73.91**	**44.88**	**68.27**
50%	LLM-Surgeon	9.43	68.77	63.22	56.19	56.19	31.83	55.24
	DISP-LLM	**7.11**	59.27	68.77	58.63	52.57	33.28	54.5
	týr-the-Pruner	9.96	59.27	68.99	58.46	50.51	30.20	53.49
	IntraSlice	7.72	**65.75**	**69.42**	**60.44**	**61.49**	**37.2**	**58.86**

Table 8: Comparison results of our method with LLM-surgeon, týr-the-Prune and DISP-LLM methods. In order to align with DISP-LLM, we only tested 5 zero-shot tasks. The test dataset of PPL is wikitext2. And the length of wikitext2 for testing is 4096.

4096-length inputs. The generation speed is evaluated with a 4096-length KV cache. Compared with SoBP(Wei et al., 2024), our method achieves comparable or superior acceleration, while also maintaining better model performance. Compared with LLM-Surgeon (van der Ouderaa et al., 2024) and DIPS-LLM (Gao et al., 2024), IntraSlice has a significant speed advantage, especially in the case of long sequences. This is because the pruning strategies of LLM-Surgeon and DIPS-LLM cannot reduce the computational complexity of attention in the MHA.

G Detailed Results

As shown in Table 8, our method demonstrates a clear advantage over DISP-LLM, týr-the-Pruner and LLM-Surgeon across all five zero-shot tasks. On average, the accuracy surpasses that of DISP-LLM by 8%. We attribute this improvement partly to DISP-LLM's end-to-end training on Wikitext2, which may lead to a certain degree of overfitting.

Detailed comparisons of model compression results are presented in Table 9 and Table 10. Our method has significant advantages over Wanda, SliceGPT and SVD-LLM. For smaller models such as LLaMA2-7B and LLaMA3-8B, IntraSlice generally outperforms SoBP, although performance varies somewhat across individual tasks. However, for larger models—including LLaMA2-13B, LLaMA2-70B, and Phi-3-Medium (14B)—IntraSlice consistently maintains a significant advantage. We believe this is because larger models contain more redundant information and have greater parameter capacity, making adaptive strategies like head pruning more effective.

Sparsity	Model	PPL	WinoGrande	PIQA	OBQA	hellaswag	BoolQ	ARC_e	ARC_c	Avg
			acc	acc-norm	acc-norm	acc-norm	acc	acc-norm	acc-norm	
0%	**Llama2-7b**	5.47	68.98	79.05	44.2	76.02	77.74	74.58	46.25	66.69
20%	SliceGPT	6.86	64.56	69.21	40.4	59.02	48.07	55.05	35.32	53.09
	Wanda	7.38	64.56	75.84	42	70.75	68.07	66.96	40.53	61.25
	SVD-LLM	8.52	60.69	66.65	36.4	54.23	49.17	50.04	30.03	49.6
	SoBP	6.51	**69.14**	76.66	40.40	65.24	**75.66**	69.28	41.89	63.54
	IntraSlice	**6.27**	68.43	**77.26**	**42.00**	**72.27**	73.12	**69.70**	**43.34**	**63.73**
30%	SliceGPT	8.62	62.59	64.69	31.8	49.12	38.9	50.17	31.14	46.92
	Wanda	9.17	57.85	72.09	39.2	62.67	63.06	60.94	38.14	56.28
	SVD-LLM	10.95	58.56	61.59	34	45.01	47.13	42.59	27.13	45.14
	SoBP	7.59	66.38	73.45	38.6	67.36	**70.92**	60.48	38.05	59.32
	IntraSlice	**7.11**	**66.69**	**74.16**	**39.6**	**67.77**	70.28	**66.29**	**38.65**	**60.49**
40%	SliceGPT	12.79	57.77	57.89	28.8	39.48	37.83	44.02	27.22	41.86
	Wanda	14.33	49.96	62.89	26	32.74	57.83	45.29	26.62	43.05
	SVD-LLM	16.58	55.72	56.2	29.8	35.55	38.99	34.81	24.49	39.37
	SoBP	9.32	**65.51**	69.26	36.40	**61.72**	56.94	55.01	35.07	54.27
	IntraSlice	**8.39**	64.09	**69.42**	**38.00**	60.86	**69.33**	**57.83**	**35.15**	**56.68**
0%	**Llama2-13b**	4.88	72.3	80.52	45.2	79.38	80.58	77.53	49.23	69.25
20%	SliceGPT	6.04	68.35	71.33	41	62.79	44.8	67.3	41.81	56.76
	Wanda	6.66	69.46	75.68	42	64.63	66.33	67.17	41.72	61
	SVD-LLM	6.78	66.54	71.6	41.8	59.82	73.88	60.4	35.92	58.56
	SoBP	5.61	**72.06**	78.45	43.8	**77.21**	78.17	**77.15**	47.53	67.77
	IntraSlice	**5.48**	71.74	**78.45**	**44.4**	76.84	**80.92**	75.72	**47.53**	**67.94**
30%	SliceGPT	7.44	65.35	65.51	39	52.33	38.9	53.16	36.77	50.15
	Wanda	10.14	57.22	53.32	35.6	47.56	62.48	31.44	25.43	44.72
	SVD-LLM	8.21	64.48	65.89	36.6	50.29	66.79	51.56	29.52	52.16
	SoBP	6.21	71.35	76.88	41.8	74.48	78.23	74.71	**47.27**	66.39
	IntraSlice	**5.96**	**71.74**	**76.88**	**45.0**	**75.04**	**79.72**	**75.21**	46.33	**67.13**
40%	SliceGPT	10.61	61.25	59.25	35.8	42.57	37.83	44.28	29.78	44.39
	Wanda	21.34	51.38	56.8	27.8	34.12	62.08	32.15	25.09	41.35
	SVD-LLM	11.26	60.06	59.3	33	40.9	52.26	40.78	25.94	44.61
	SoBP	7.32	67.88	67.14	38.0	68.49	74.62	53.58	34.73	57.78
	IntraSlice	**6.92**	**68.27**	**72.69**	**40.6**	**69.01**	**78.17**	**64.94**	**41.98**	**62.24**
0%	**Llama2-70b**	3.32	77.98	82.75	48.8	83.81	83.70	80.98	57.25	73.61
20%	Wanda	4.1	75.77	80.25	46.8	81.28	81.68	77.06	53.16	70.86
	SoBP	3.91	76.80	81.50	**49.4**	82.97	68.81	**80.81**	**58.02**	71.19
	IntraSlice	**3.85**	**76.95**	**81.88**	48.0	**83.71**	**84.16**	79.80	56.14	**72.95**
30%	Wanda	4.77	75.22	79.27	45.4	79.1	81.5	75.84	52.99	69.9
	SoBP	4.41	76.48	80.36	47.6	81.57	67.98	77.95	53.84	69.40
	IntraSlice	**4.34**	**76.87**	**80.63**	**47.8**	**82.33**	**85.02**	**79.04**	**54.18**	**72.27**
40%	Wanda	4.77	75.22	79.27	45.4	79.1	81.5	75.84	52.99	69.9
	SoBP	4.99	76.16	78.89	**47.6**	79.49	71.53	**75.34**	**50.60**	68.51
	IntraSlice	**4.91**	**76.40**	**79.27**	46.0	**79.49**	**84.89**	73.95	49.91	**69.99**

Table 9: Detail comparison of model compression results on Llama2-7b, Llama2-13b and Llama2-70b. PPL is the result on wikitext2.

Sparsity	Model	PPL	WinoGrande	PIQA	OBQA	hellaswag	BoolQ	ARC_e	ARC_c	Avg
			acc	acc-norm	acc-norm	acc-norm	acc	acc-norm	acc-norm	
0%	**Llama3-8b**	6.13	72.77	80.74	45.0	79.13	81.50	77.74	53.58	70.07
20%	SliceGPT	10.93	62.9	63.17	34.4	52.29	38.13	52.95	32.85	48.1
	Wanda	122.41	49.49	53.54	25.4	30.77	50.7	29.67	23.12	37.53
	SVD-LLM	47	53.51	62.79	30.8	40.93	58.47	45.2	26.28	45.43
	SoBP	8.74	**71.82**	75.73	**41.6**	71.00	68.07	**73.02**	43.69	63.56
	IntraSlice	**8.27**	70.48	**76.39**	41.4	**71.30**	**78.96**	72.64	**45.31**	**65.21**
30%	SliceGPT	17.02	57.7	56.26	31	39.67	37.83	41.08	26.28	41.4
	Wanda	271.71	48.38	52.61	25.8	28.46	50.15	29.25	20.82	36.5
	SVD-LLM	101.56	51.85	57.78	27.2	32.94	58.69	34.97	22.18	40.8
	SoBP	10.32	**70.09**	72.63	**39.8**	64.19	65.17	61.87	36.43	58.60
	IntraSlice	**10.25**	67.32	**73.39**	39.0	**64.20**	**70.86**	**67.63**	**42.15**	**60.65**
40%	SliceGPT	30.8	52.96	53.48	27.2	32.41	37.83	35.35	22.53	37.39
	Wanda	4258.41	50.51	50.49	24.2	26.61	41.41	26.14	23.38	34.68
	SVD-LLM	207.99	50.51	54.52	26	29.71	43.58	31.1	21.59	36.71
	SoBP	12.48	**65.90**	68.12	**35.4**	**54.47**	**62.26**	53.16	29.69	**52.71**
	IntraSlice	**12.28**	61.88	**68.82**	35.00	51.89	51.65	**56.10**	**32.34**	51.10
0%	**Phi-3-medium-4k**	4.29	76.56	81.61	50.8	82.73	88.53	81.31	61.69	74.75
20%	SliceGPT	6.45	74.03	73.5	44.4	67.41	47.16	75.59	52.05	62.02
	Wanda	6.82	70.32	77.69	46	77.48	84.34	77.82	52.99	69.52
	SVD-LLM	7.16	74.11	76.93	44.4	70.71	82.63	75.97	51.37	68.02
	SoBP	6.27	73.48	80.09	**47.2**	76.96	**88.04**	82.24	58.02	72.29
	IntraSlice	**5.80**	**73.72**	**80.14**	46.8	**79.13**	87.77	**84.34**	**59.56**	**73.06**
30%	SliceGPT	7.66	68.35	66.81	38.6	56.31	56.91	62.79	41.98	55.97
	Wanda	10	61.88	74.37	41.2	70.39	69.69	71	48.29	62.41
	SVD-LLM	8.22	71.67	72.09	38.8	61.01	79.36	65.7	42.75	61.62
	SoBP	7.05	71.43	75.19	43.8	72.75	87.31	73.15	48.98	67.52
	IntraSlice	**6.71**	**72.69**	**77.15**	**45.8**	**74.07**	**86.12**	**79.12**	**53.16**	**69.73**
40%	SliceGPT	10.01	63.22	61.59	34.8	44.72	37.89	43.9	30.03	45.16
	Wanda	20.68	61.17	71.22	40	59.01	41.41	61.07	40.78	53.52
	SVD-LLM	10.76	64.25	62.35	37.2	48.82	67.55	53.87	33.53	52.51
	SoBP	8.02	**69.77**	69.42	**42.4**	65.68	**86.09**	54.76	39.93	61.15
	IntraSlice	**7.90**	67.88	**75.08**	41.4	**66.38**	84.07	**72.73**	**46.76**	**64.90**

Table 10: Detail comparison of model compression results on Llama3-8b and Phi-3-medium-4k. PPL is the result on wikitext2.