Beyond Uniform SVD: Dual-Level Optimization across Columns and Modules for LLM Compression

Lin Xv[1], Xian Gao[1], Ting Liu[1,*], Yuzhuo Fu[1,*]
[1]Shanghai Jiao Tong University
xv-lin@sjtu.edu.cn

Abstract

Low-rank decomposition, particularly Singular Value Decomposition (SVD), is a pivotal technique for mitigating the storage and computational demands of Large Language Models (LLMs). However, prevalent SVD-based approaches overlook the critical phenomenon that decomposition errors exhibit significant disparity across different components of the parameter matrix, often leading to suboptimal approximation. Furthermore, existing methods lack a direct metric to evaluate the importance of individual weight matrices. To address these limitations, we propose **Duo-SVD** (**Du**al-level **O**ptimization **SVD**), a novel training-free framework that synergizes optimization at both the column and the module levels. First, Duo-SVD incorporates a *Column-Preserving Strategy* that explicitly retains columns exhibiting high decomposition errors, while applying low-rank approximation solely to those with lower errors. Second, at the module level, we employ a *Module-Adaptive Allocation Strategy* that formulates ratio allocation as a global constrained optimization problem based on perturbation-induced model deviation. Extensive experiments demonstrate that Duo-SVD consistently outperforms state-of-the-art SVD-based baselines and structured pruning methods, establishing it as a superior paradigm for efficient LLM compression. Code is available at https://github.com/Pupu792/Duo-SVD.

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Yang et al., 2025) have revolutionized Natural Language Processing, yet their escalating parameter scales—ranging from tens to hundreds of billions—pose severe deployment challenges (Zhou et al., 2024; Chavan et al., 2024). To address this, model compression techniques such as Quantization (Frantar et al., 2023; LinJi et al., 2025), Pruning (Ma et al., 2023; Yang et al., 2024), Knowledge Distillation (Gu et al., 2024; Zhong et al., 2024), and Low-Rank Decomposition (Ashkboos et al., 2024; Lin et al., 2025) have been widely adopted.

Among these, Singular Value Decomposition (SVD) offers a theoretically grounded approach for compression (Eckart and Young, 1936). However, most existing SVD-based methods (Yuan et al., 2024; Wang et al., 2025a; Qinsi et al., 2025) apply a holistic decomposition strategy. Although approaches like SoLA (Huang et al., 2025) leverage activation sparsity to retain dominant channels, they still lack a deeper exploration into the intrinsic properties of the weight matrices themselves.

In this regard, we identify a critical phenomenon: weight matrices exhibit *structural heterogeneity* concerning SVD decomposition. Specifically, we observe that decomposition errors are non-uniformly distributed: certain columns of the matrix yield significant reconstruction errors, whereas errors in other columns are negligible. This implies that a uniform decomposition strategy, which overlooks such structural heterogeneity, leads to suboptimal compression performance.

Beyond the internal characteristics of weight matrices, another widely recognized phenomenon is the heterogeneous importance across different modules within LLMs (Yin et al., 2024; Lu et al., 2024). This necessitates varying rank retention for different modules. Prior works attempt to solve this via varying heuristics: Bolaco (Ji et al., 2024) employs Bayesian optimization, SVD-LLM v2 (Wang et al., 2025b) relies on truncation error thresholds, and SoLA (Huang et al., 2025) minimizes module-wise truncation errors. While effective to a degree, these methods rely on proxy metrics rather than a detailed assessment of each module's impact on the model's actual performance, leaving significant room for optimization.

To bridge these gaps, we propose **Duo-SVD**

*Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 18335–18349

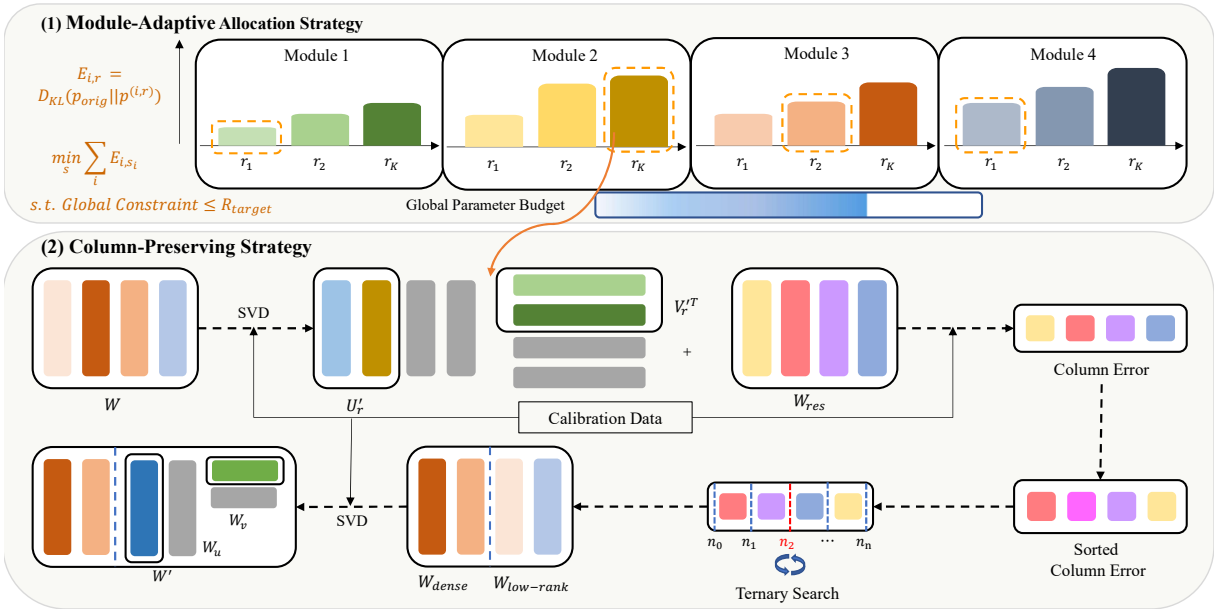


Figure 1: Framework of Duo-SVD. (1) Compute the perturbation error of each module and solve for the module-wise compression ratios that minimize the overall perturbation error. (2) Compute the residual matrices and column-wise errors for each module, then determine which columns to retain and decompose the remaining columns.

(**Du**al-level **O**ptimization **SVD**), a novel training-free framework that synergizes optimization at both the column and module levels.

First, targeting micro-level *structural heterogeneity*, Duo-SVD incorporates a **Column-Preserving Strategy**. By leveraging the varying sensitivity of different weight columns, Duo-SVD explicitly retains high-sensitivity columns in their dense form while applying low-rank decomposition to the remaining ones. Furthermore, by exploiting the monotonicity of the decomposition error, it efficiently determines the optimal number of retained columns. Second, regarding macro-level resource distribution, we introduce a **Module-Adaptive Allocation Strategy**. By evaluating the impact of each module on the model's output under varying compression ratios, we formulate the compression ratio allocation as a global constrained optimization problem. This allows the framework to automatically identify the optimal compression configuration that minimizes the total errors under a strict global parameter budget.

Our key contributions are summarized as follows:

- We identify the limitation of uniform SVD in handling the structural heterogeneity of weight matrices and propose a *Column-Preserving Strategy* that integrates exact column retention with low-rank approximation to reduce reconstruction error.
- We formulate rank allocation as a global constrained optimization problem, introducing a *Module-Adaptive Allocation Strategy* to precisely manage the compression budget across modules based on perturbation sensitivity.
- Extensive experiments demonstrate that Duo-SVD consistently outperforms state-of-the-art SVD baselines and structured pruning methods on both language modeling and downstream tasks, marking a significant advancement in compression efficacy.

2 Related Work

Pruning. Pruning methods are categorized into unstructured and structured approaches. Unstructured pruning, such as SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2024), and BESA (Xu et al., 2024), operates on individual weights, removing parameters based on importance metrics like Hessian information or activation magnitude. Although effective in preserving model performance, these methods struggle to achieve practical acceleration on GPUs due to irregular sparsity patterns. Conversely, structured pruning reduces model size by removing parameter channels or layers, which is more hardware-friendly. For instance, LLM-Pruner (Ma et al., 2023) leverages gradient information to remove coupled structures, FLAP (An et al., 2024) substitutes stable activation pat-

terns with bias terms, and methods like ShortGPT (Men et al., 2025) and BlockPruner (Zhong et al., 2025) identify depth redundancy to prune entire layers.

Low-Rank Decomposition. Decomposition techniques can be divided into single-module and module-pair approaches. For single-module decomposition, Standard SVD is typically refined to mitigate accuracy loss: FWSVD (Hua et al., 2022) incorporates Fisher information, while ASVD (Yuan et al., 2024) scales weight matrices based on input activations. Recognizing that activations often exhibit stronger low-rank characteristics than weights, methods such as AFM (Yu and Wu, 2023), Bolaco (Ji et al., 2024), and Dobi-SVD (Qinsi et al., 2025) perform PCA or its variants on activations to derive column-orthogonal matrices for compression. Other works like SVD-LLM (Wang et al., 2025c) directly analyze the weight-activation product, while Basis Sharing (Wang et al., 2025a) further improves compression by sharing low-rank matrices across layers. Notably, SoLA (Huang et al., 2025) adopts a decoupling strategy to explicitly retain dominant MLP channels. Regarding module-pair decomposition, methods like SliceGPT (Ashkboos et al., 2024) and MoDeGPT (Lin et al., 2025) treat two adjacent matrices as a unified block, achieving compression by jointly reducing their intermediate dimension.

3 Preliminaries

Singular Value Decomposition. For a weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ with $m \leq n$ and an input calibration dataset $\mathbf{X} \in \mathbb{R}^{n\times d}$, the compressed representation is formulated as $\mathbf{W}' = \mathbf{U}'_r {\mathbf{V}'_r}^\top$, where $\mathbf{U}'_r \in \mathbb{R}^{m\times r}$ and ${\mathbf{V}'_r}^\top \in \mathbb{R}^{r\times n}$. The objective is to minimize the reconstruction error under a rank constraint:

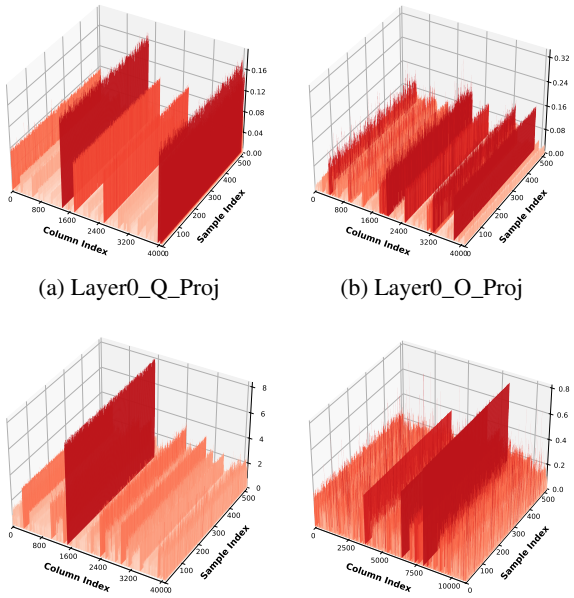
$$\begin{aligned} \min_{\mathbf{W}'} \quad & L_{\text{SVD},r} = \|\mathbf{WX} - \mathbf{W}'\mathbf{X}\|_F \\ \text{s.t.} \quad & \text{rank}(\mathbf{W}') \leq r. \end{aligned} \tag{1}$$

Following the formulation in (Wang et al., 2025c), let $\mathbf{H} = \mathbf{X}\mathbf{X}^\top$ and let $\mathbf{S}$ be its Cholesky factor such that $\mathbf{H} = \mathbf{S}\mathbf{S}^\top$. We apply SVD to the whitened weight matrix $\mathbf{WS} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Here, $\mathbf{\Sigma} = \text{diag}(\delta_1, \ldots, \delta_m)$ contains singular values sorted in descending order ($\delta_1 \geq \cdots \geq \delta_m$). By retaining the top-$r$ singular values and their corresponding vectors, we obtain the truncated approximation $\mathbf{W} \approx \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^\top\mathbf{S}^{-1}$. This allows us to decompose the original weight into two low-rank factors: $\mathbf{U}'_r = \mathbf{U}_r\sqrt{\mathbf{\Sigma}_r}$ and ${\mathbf{V}'_r}^\top = \sqrt{\mathbf{\Sigma}_r}\mathbf{V}_r^\top\mathbf{S}^{-1}$. The resulting truncation loss is determined by the discarded singular values:

$$L^2_{\text{SVD},r} = \sum_{i=r+1}^{m} \delta_i^2. \tag{2}$$

Cauchy Interlacing Theorem. Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ (where $m \leq n$) be a matrix with singular values $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_m(\mathbf{A})$. If $\mathbf{B}$ is a submatrix constructed by removing k columns from $\mathbf{A}$, then for every $j \in \{1, \ldots, m-k\}$, the singular values of $\mathbf{B}$ satisfy the following interlacing property:

$$\sigma_j(\mathbf{A}) \geq \sigma_j(\mathbf{B}) \geq \sigma_{j+k}(\mathbf{A}). \tag{3}$$



(a) Layer0_Q_Proj (b) Layer0_O_Proj

(c) Layer0_Gate_Proj (d) Layer0_Down_Proj

Figure 2: Column Errors in LLaMA-2 7B

4 Method

We present **Duo-SVD**, a dual-level optimization framework, as shown in Figure 1. In Section 4.1, we first introduce the *Column-Preserving Strategy* to address structural heterogeneity within weight matrices. Subsequently, we detail the *Module-Adaptive Allocation Strategy*, which optimizes the global compression budget across different weight matrices, in Section 4.2. The complete procedure is summarized in Algorithm 1.

4.1 Column-Preserving Strategy

Motivation: structural Heterogeneity. Holistic SVD approaches minimize the global reconstruc-

tion error $L_{\text{SVD}} = \|\mathbf{W}\mathbf{X} - \mathbf{W}'\mathbf{X}\|_F$. However, this global objective obscures the *structural heterogeneity* inherent in LLM weight matrices: the approximation error is often highly non-uniform across weight columns. To quantify this, let $\mathbf{W}_{\text{res}} = \mathbf{W} - \mathbf{W}'$ denote the residual matrix arising from SVD truncation. We define the truncation error for the j-th column, denoted as $L_{\text{col},j}$, by projecting the column-wise residual onto the input activation space:

$$L_{\text{col},j} = \|\mathbf{W}_{\text{res}}[:, j] \cdot \mathbf{X}[j, :]\|_F, \tag{4}$$

where $\mathbf{W}_{\text{res}}[:, j]$ is the j-th column of the residual matrix and $\mathbf{X}[j, :]$ is the corresponding row of the input $\mathbf{X}$. Empirical observations (as illustrated in Figure 2) reveal that $\{L_{\text{col},j}\}$ exhibits a structurally concentrated distribution. The truncation error is dominated by a sparse subset of highly sensitive columns that remain consistent across different inputs. Consequently, uniformly compressing these sensitive columns leads to significant accuracy loss.

Hybrid Decomposition and Computation. To mitigate structural heterogeneity, we propose a hybrid compression strategy that partitions a weight matrix into a dense component and a low-rank component. Let $\mathcal{S} \subset \{1, \ldots, n\}$ denote the index set of columns to be preserved exactly, with cardinality $c = |\mathcal{S}|$, and let $\bar{\mathcal{S}}$ denote the complement set for compression. Accordingly, the weight matrix $\mathbf{W}$ is split into $\mathbf{W}_{\mathcal{S}} \in \mathbb{R}^{m \times c}$ and $\mathbf{W}_{\bar{\mathcal{S}}} \in \mathbb{R}^{m \times (n-c)}$ and we apply low-rank decomposition solely to $\mathbf{W}_{\bar{\mathcal{S}}}$, approximating it as $\mathbf{W}_u \mathbf{W}_v$, where $\mathbf{W}_u \in \mathbb{R}^{m \times r}$ and $\mathbf{W}_v \in \mathbb{R}^{r \times (n-c)}$.

Correspondingly, the forward propagation $\mathbf{Y} = \mathbf{W}'\mathbf{X}$ is computationally decoupled into a dual-pathway formulation. Let $\mathbf{X}_{\mathcal{S}}$ and $\mathbf{X}_{\bar{\mathcal{S}}}$ represent the subsets of input rows corresponding to the preserved and compressed indices, respectively. The output is computed as:

$$\mathbf{W}'\mathbf{X} = \underbrace{\mathbf{W}_{\mathcal{S}}\mathbf{X}_{\mathcal{S}}}_{\text{Dense Path}} + \underbrace{\mathbf{W}_u(\mathbf{W}_v\mathbf{X}_{\bar{\mathcal{S}}})}_{\text{Low-Rank Path}}. \tag{5}$$

This formulation ensures that high-sensitivity columns are processed with full precision via the dense path, while the remaining columns are projected through a parameter-efficient bottleneck.

Our goal is to determine the cardinality c and the optimal subset $\mathcal{S}$ that minimize the reconstruction error under a parameter budget B. Formally, the optimization problem is:

$$\begin{aligned} \min_{c,\mathcal{S}} \quad & \mathcal{F}(c, \mathcal{S}) = \|\mathbf{W}\mathbf{X} - \mathbf{W}'(\mathcal{S}, r)\mathbf{X}\|_F^2 \\ \text{s.t.} \quad & \mathcal{S} \subset \{1, \ldots, n\}, \quad |\mathcal{S}| = c, \\ & mc + r(m + n - c) \leq B. \end{aligned} \tag{6}$$

Here, mc represents the cost of preserved columns, and $r(m + n - c)$ represents the cost of the SVD factorization for the remaining columns. Note that under a fixed budget B, the rank r is inversely constrained by c: preserving more dense columns reduces the budget available for the low-rank path, necessitating a lower rank r.

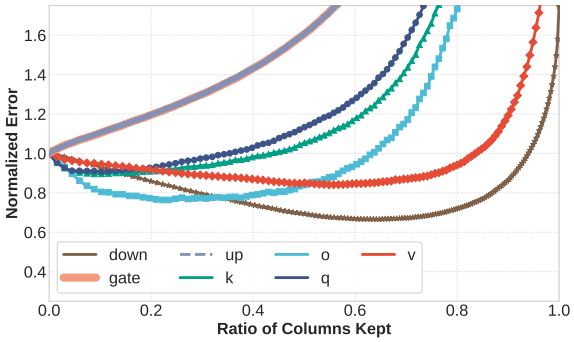


Figure 3: $\mathcal{F}(c)$ for LLaMA-2 7B Layer 0 at 20% Compression Ratio.

Efficient Solver: Greedy Selection and Unimodality Analysis. Solving Eq. 6 involves a combinatorial search over $\mathcal{S}$ and c, which is typically intractable. We decompose this into two manageable sub-problems.

Step 1: Optimal Subset Selection. For a fixed cardinality c, we aim to find the optimal subset $\mathcal{S}_c^*$. Adopting a *Column Independence Assumption*, we evaluate column importance independently, ignoring inter-column correlations. Under this approximation, the total error reduction decouples into a sum of independent column error terms, simplifying the strategy to a deterministic *Greedy Selection*. By sorting the column errors in descending order $L_{\text{col},(1)} \geq L_{\text{col},(2)} \geq \cdots \geq L_{\text{col},(n)}$, the optimal subset is simply the top-c columns: $\mathcal{S}_c^* = \{(1), \ldots, (c)\}$. This reduces the objective function to a univariate function of c, denoted as $\mathcal{F}(c)$.

Step 2: Optimal Cardinality Search. Exact evaluation is computationally intractable as it necessitates a fresh SVD calculation for the complementary matrix at every candidate c. Instead, we analyze the discrete gradient $\Delta\mathcal{F}(c)$ over a step size

k, where k is the minimal increment in preserved columns that reduces the allowable rank by one unit i.e., $r(c+k) = r(c) - 1$. Let $\mathbf{W}_{c+k}$ and $\mathbf{W}_c$ denote the complementary matrices. Given $m \leq n - c$, the error difference is derived as:

$$\begin{aligned}
&\Delta\mathcal{F}(c) \\
&= L^2_{\text{SVD},r(c)-1}(\mathbf{W}_{c+k}) - L^2_{\text{SVD},r(c)}(\mathbf{W}_c) \\
&= \sum_{i=r(c)}^{m} \sigma_i^2(\mathbf{W}_{c+k}) - \sum_{i=r(c)+1}^{m} \sigma_i^2(\mathbf{W}_c) \\
&= \sigma^2_{r(c)}(\mathbf{W}_{c+k}) \\
&\quad + \underbrace{\sum_{i=r(c)+1}^{m} \left(\sigma_i^2(\mathbf{W}_{c+k}) - \sigma_i^2(\mathbf{W}_c)\right)}_{\leq 0 \text{ (Eq. 3)}} \\
&\approx \underbrace{\sigma^2_{r(c)}(\mathbf{W}_{c+k})}_{\leq \delta^2_{r(c)}(\mathbf{W}) \text{ (Eq. 3)}} - \sum_{j=1}^{k} L^2_{\text{col},(c+j)} \\
&\leq \delta^2_{r(c)}(\mathbf{W}) - \sum_{j=1}^{k} L^2_{\text{col},(c+j)}.
\end{aligned} \tag{7}$$

The approximation substitutes the spectral energy loss with the loss of the newly preserved columns, i.e., $-\sum L^2_{\text{col}}$. Consequently, the upper bound of $\Delta\mathcal{F}(c)$ can be viewed as a composition of two competing components: a *cost term* $\delta^2_{r(c)}(\mathbf{W})$ induced by the rank reduction, and a *gain term* $\sum_{j=1}^{k} L^2_{\text{col},(c+j)}$ derived from column preservation.

As c increases, the rank $r(c)$ decreases, shifting the truncation index toward larger singular values, which causes the cost term to monotonically increase. Conversely, the greedy selection strategy ensures that the gain term exhibits a decreasing trend. The superposition of an increasing cost and a decreasing gain implies that the discrete gradient of the derived upper bound is increasing, establishing the convexity of the approximated objective. Although based on approximation, empirical observations confirm that the exact function $\mathcal{F}(c)$ consistently exhibits unimodality (see Figure 3 and further verification in Appendix B). This alignment justifies the use of Ternary Search to efficiently locate c^* with logarithmic complexity. Note that a similar analysis for $m \geq n - c$ is provided in Appendix C.

4.2 Module-Adaptive Allocation Strategy

While the Column-Preserving Strategy optimizes within a matrix, LLMs exhibit significant sensitivity variance across different modules. To address this, we propose a perturbation-based allocation scheme.

Sensitivity Metric Definition. Consider a model with M modules $\{\mathcal{M}_1, \ldots, \mathcal{M}_M\}$ and parameter counts $\{P_1, \ldots, P_M\}$. Let $\mathcal{R} = \{r_1, \ldots, r_K\}$ be a discrete set of candidate compression ratios. We define the sensitivity of module i at a candidate ratio $r \in \mathcal{R}$ by measuring the divergence between the original model output distribution p_{orig} and the perturbed output $p^{(i,r)}$.

Specifically, $p^{(i,r)}$ represents the output distribution obtained by compressing module i to ratio r while maintaining all other modules at the global target rate R_{target}. The corresponding sensitivity error is quantified as:

$$E_{i,r} = D_{\text{KL}}(p_{\text{orig}} \| p^{(i,r)}), \quad \forall r \in \mathcal{R}. \tag{8}$$

Global Optimization via Integer Programming. Our objective is to determine the optimal ratio configuration vector $\mathbf{s} = [s_1, \ldots, s_M]$, where each $s_i \in \mathcal{R}$ denotes the selected compression ratio for module i. The goal is to minimize the aggregate sensitivity error while strictly adhering to the global parameter budget. This is formulated as a Constrained Optimization Problem:

$$\begin{aligned}
\mathbf{s}^* = \arg\min_{\mathbf{s} \in \mathcal{R}^M} \quad & \sum_{i=1}^{M} E_{i,s_i} \\
\text{s.t.} \quad & \frac{\sum_{i=1}^{M} s_i P_i}{\sum_{i=1}^{M} P_i} \leq R_{\text{target}}.
\end{aligned} \tag{9}$$

This formulation is structurally equivalent to the *Multiple-Choice Knapsack Problem* (MCKP). Given that M is finite and the candidate set $\mathcal{R}$ is small, we employ a Dynamic Programming approach to efficiently solve for the globally optimal allocation $\mathbf{s}^*$.

5 Experiments

5.1 Experimental Setup

Baselines. We compare Duo-SVD with structured pruning methods, including LLM-Pruner (Ma et al., 2023) and FLAP (An et al., 2024), as well as various decomposition techniques such as SliceGPT (Ashkboos et al., 2024), Bolaco (Ji et al., 2024), SVD-LLM (Wang et al., 2025c), and SoLA (Huang et al., 2025). Furthermore, we investigate the combination of Duo-SVD with quantization, comparing it against quantization-only

Model	Method	PPL ↓	Avg. ↑	MMLU-5shot	PIQA	WinoG.	HellaS.	ARC-e	ARC-c	OBQA
LLaMA-2 7B	Dense	5.11	62.14	45.70	79.05	69.38	75.92	74.49	46.25	44.20
	LLM-Pruner	10.55	53.89	26.20	75.95	63.38	67.83	64.31	39.93	39.60
	FLAP	6.76	53.07	31.90	74.54	62.98	64.74	61.28	36.43	39.60
	SliceGPT	8.24	46.26	26.75	64.80	62.98	49.18	55.68	31.40	33.00
	SVD-LLM	7.84	45.59	26.80	65.13	62.43	51.73	47.22	27.82	38.00
	Bolaco	7.31	55.23	34.30	75.09	65.61	64.33	68.19	37.48	41.60
	SoLA	6.52	54.33	34.10	74.65	**66.46**	63.92	65.61	37.37	38.20
	Duo-SVD	**5.99**	**57.27**	**35.41**	**76.77**	65.67	**70.23**	**70.50**	**40.53**	**41.80**
LLaMA-2 13B	Dense	4.57	65.70	55.40	80.41	72.53	79.41	77.39	49.15	45.60
	LLM-Pruner	9.67	55.45	22.80	**77.97**	60.77	71.26	67.09	44.28	44.00
	FLAP	5.90	57.00	41.20	75.57	67.25	69.19	65.91	39.08	40.80
	SliceGPT	7.10	50.58	35.49	65.18	65.67	52.30	59.26	36.77	39.40
	SVD-LLM	7.37	53.39	34.60	71.60	68.43	59.91	62.12	36.69	40.40
	Bolaco	6.32	59.22	43.40	76.83	65.90	69.96	70.93	42.72	**44.80**
	SoLA	5.61	58.83	46.10	75.57	69.77	67.35	69.15	40.70	43.20
	Duo-SVD	**5.12**	**63.18**	**48.73**	**77.97**	**71.03**	**77.40**	**75.00**	**47.35**	**44.80**
Mistral -7B	Dense	4.92	68.14	62.50	82.05	73.95	81.02	79.55	53.92	44.00
	FLAP	7.11	48.29	25.90	72.31	64.09	55.94	51.05	31.91	36.80
	SliceGPT	9.06	43.18	25.52	59.35	61.21	45.11	51.60	30.29	29.20
	SVD-LLM	9.29	50.23	25.02	70.46	64.56	58.09	69.28	37.63	26.60
	SoLA	6.06	56.98	44.20	73.67	**68.75**	63.32	**69.99**	39.76	**39.20**
	Duo-SVD	**5.78**	**59.37**	**47.74**	**78.67**	66.61	**72.20**	69.15	**43.43**	37.80
LLaMA-3.1 8B	Dense	5.84	68.36	65.32	81.18	73.64	78.89	81.10	53.41	45.00
	LLM-Pruner	15.99	50.12	28.15	**72.69**	58.09	56.75	62.50	36.86	35.80
	FLAP	9.23	51.52	37.85	72.14	63.69	59.04	56.31	32.00	39.60
	SVD-LLM	13.72	44.46	26.51	62.57	60.77	49.93	48.02	31.23	32.20
	Duo-SVD	**8.78**	**57.80**	**50.55**	72.63	**70.24**	**65.91**	**65.61**	**38.82**	**40.80**

Table 1: Comparison of WikiText-2 perplexity and downstream task accuracy of different methods on LLaMA-2 7B/13B, Mistral-7B, and LLaMA-3.1 8B models at 20% compression ratio.

baselines. Additional comparisons with BasisSharing (Wang et al., 2025a), Dobi-SVD (Qinsi et al., 2025), Laco (Yang et al., 2024), ShortGPT (Men et al., 2025), and BlockPruner (Zhong et al., 2025) are provided in Appendix D.

Models and Datasets. We conduct a comprehensive evaluation on the widely studied LLaMA-2 7B/13B models (Touvron et al., 2023). To further verify the generalization capabilities of our method, we extend our experiments to the Mistral-7B (Jiang et al., 2023) and LLaMA-3.1 8B (Grattafiori et al., 2024) models. We assess model performance across both language modeling and downstream tasks. For language modeling, we utilize Perplexity (PPL) on WikiText-2 (Merity et al., 2017) datasets as the primary metric. For downstream tasks, we evaluate the model's comprehensive learning capability using 5-shot MMLU (Hendrycks et al., 2021) and assess zero-shot reasoning performance across OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-e, ARC-c (Clark et al., 2018), and PIQA (Bisk et al., 2020).

Implementation Details. Regarding calibration data, following the protocols of SVD-LLM, we randomly select 256 samples from Wikitext2, each containing 2048 tokens. For the MAAS phase, consistent with Section 4.2, the candidate compression ratio set $\mathcal{R}$ consists of 10 elements ranging from 0 to 0.9 (i.e., $\mathcal{R} = \{0, 0.1, \ldots, 0.9\}$), and the module-wise sensitivity $E_{i,s}$ is computed using 32 samples, each with 2048 tokens from Wikitext2. All experiments are conducted on NVIDIA RTX 3090 GPUs using the Hugging Face Transformers library, with downstream tasks evaluated via the LM-Evaluation-Harness (Gao et al., 2021).

5.2 Experimental Results

Performance Comparison. Table 1 summarizes the quantitative results across LLaMA-2 7B/13B, Mistral-7B, and LLaMA-3.1 8B models. As observed, our proposed Duo-SVD consistently outperforms state-of-the-art baselines, achieving the lowest perplexity and the highest average accuracy across all evaluated architectures. For instance, on LLaMA-2 13B, Duo-SVD maintains an impressive average accuracy of 63.18% with a minimal perplexity of 5.12, effectively preserving the model's

generation capabilities. Furthermore, in challenging reasoning benchmarks such as MMLU, our method demonstrates superior robustness, indicating that Duo-SVD effectively retains critical knowledge during compression. Overall, compared to other decomposition-based methods that often suffer from severe performance degradation on diverse model families, Duo-SVD establishes a new state-of-the-art in balancing sparsity and performance.

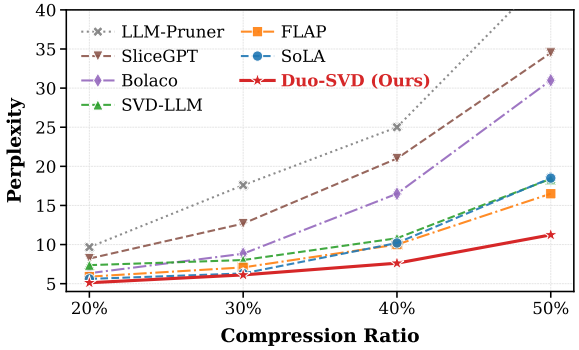


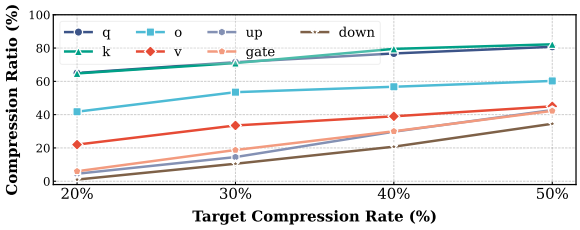
Figure 4: Comparison of WikiText-2 perplexity across different compression ratios on LLaMA-2 13B.

Model	Method	PPL↓
LLaMA-2 7B	Dense	5.12
	GPTQ 3-bit	8.05
	Duo-SVD 25%+ GPTQ 4-bit	**7.73**
LLaMA-3.1 8B	Dense	5.84
	GPTQ 3-bit	19.28
	Duo-SVD 25%+ GPTQ 4-bit	**13.63**

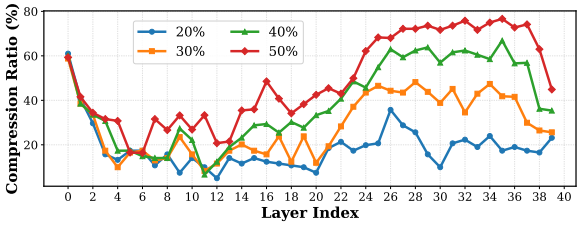
Table 2: Comparison of WikiText-2 perplexity for LLaMA-2 7B and LLaMA-3.1 8B models using Duo-SVD and GPTQ.

To further evaluate robustness under aggressive compression, Figure 4 illustrates the perplexity trends on LLaMA-2 13B across compression ratios from 20% to 50%. While baselines such as LLM-Pruner and SliceGPT exhibit rapid performance deterioration as sparsity increases, Duo-SVD demonstrates a significantly flatter growth curve. Even at 50% ratio, Duo-SVD maintains a stable PPL of 11.24, remarkably lower than SoLA and FLAP, indicating superior preservation of linguistic capabilities.

Combination with Quantization. Quantization is a crucial compression method orthogonal to decomposition. We applied 4-bit GPTQ (Frantar et al., 2023) to LLaMA-2 7B and LLaMA-3.1 8B compressed by 25% using Duo-SVD, omitting the column-preserving strategy for better integration with quantization, and compared them against pure quantization methods with the same compression ratios. The results in Table 2 indicate that, under the same compression ratios, the combination of Duo-SVD and quantization achieves lower perplexity.



(a) Average compression ratio across different module types



(b) Per-layer compression ratios under different overall compression levels

Figure 5: Allocation results of LLaMA-2 13B under different compression ratios

Results of the Module-Adaptive Allocation Strategy. Figure 5a presents the average compression ratios for different module types in the LLaMA2-13B model. It is evident that there are significant disparities in compression ratios across module types. Specifically, the `down_proj` modules consistently maintain a compression ratio lower than the overall target, whereas the `q_proj` and `k_proj` modules undergo substantial parameter reduction. Additional model results and the comparison with other methods are presented in Appendix E and Appendix F, respectively.

Furthermore, Figure 5b illustrates distinct variations in compression ratios across different layers. In general, layers in the first half of the model exhibit lower compression ratios, while those in the second half demonstrate higher ratios, with the first and last layers being notable exceptions. These results provide new insights into the varying importance of different layers under SVD decomposition.

Compression Ratio	TTFT (s) ↓	Throughput (tokens/s) ↑	Memory (GB) ↓
Dense	1.3580	557.15	19.88
20%	1.2073	579.24	17.67
30%	1.1221	610.22	16.46
40%	1.0057	625.90	15.26
50%	0.9262	642.03	14.06

Table 3: LLaMA-2 7B inference efficiency at different compression ratios. Measured with batch size 24, prefill length 256, and generation length 64.

Inference Efficiency. Based on the analysis in Appendix G, a model compressed with a ratio of r theoretically reduces both model size and GEMM computational cost. Here, we further validate the practical performance of Duo-SVD in real-world deployment scenarios. Table 3 presents the prefilling latency (Time to First Token, TTFT), autoregressive decoding throughput, and peak memory usage of the LLaMA-2 7B model. In the prefilling phase, Duo-SVD reduces inference latency by lowering computational overhead, while in the autoregressive decoding phase, it improves throughput by reducing parameter count and memory access. The results demonstrate that as the compression ratio increases, Duo-SVD not only significantly reduces memory footprint but also accelerates both the prefilling and decoding stages, highlighting its promising potential for efficient deployment.

5.3 Ablation Study

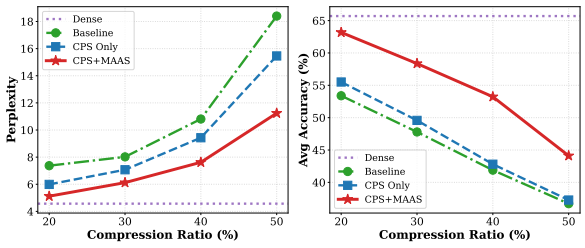


Figure 6: Impact of the MAAS on the performance of LLaMA-2 13B.

Effectiveness of Column-Preserving Strategy and Module-Adaptive Allocation Strategy. To evaluate the effectiveness of our proposed Column-Preserving Strategy (CPS) and Module-Adaptive Allocation Strategy (MAAS), we compare the performance of CPS alone and the combination of CPS and MAAS against the baseline SVD-LLM. As shown in Figure 6, compared to the baseline, employing CPS reduces perplexity by approximately 10% and increases average accuracy by about 1.5 percentage points. Furthermore, incorporating MAAS leads to an additional reduction in perplexity of nearly 10% and a significant improvement in accuracy of roughly 7 percentage points. These results demonstrate the benefits of preserving crucial columns and underscore the significant variance in sensitivity among different modules within LLMs. Consequently, leveraging these disparities to guide the compression process is essential for enhancing the performance of compressed models.

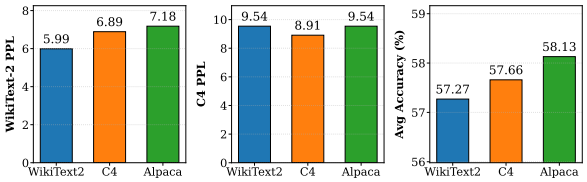


Figure 7: Performance comparison of LLaMA-2 7B at 20% compression ratio using different calibration datasets.

Dataset Robustness. Currently, we utilize the WikiText-2 dataset for CPS and MAAS. To investigate the impact of different datasets on Duo-SVD, we replaced WikiText-2 with the C4 (Raffel et al., 2020) and Alpaca (Taori et al., 2023) datasets and re-executed the compression pipeline. Figure 7 illustrates the perplexity on WikiText-2 and C4, as well as the average accuracy on downstream tasks across different calibration datasets. The calibration dataset consistently achieves better performance on its corresponding test metric. Specifically, calibrating with WikiText-2 reduces the perplexity on WikiText-2 by 0.9 compared to calibrating with C4, whereas the perplexity on C4 increases by 0.63. Regarding the average accuracy on downstream tasks, the variation caused by different datasets remains within 1%. This indicates that switching calibration datasets has a minimal impact on the performance of Duo-SVD in practical tasks, demonstrating the robustness of our method.

6 Conclusion

We proposed **Duo-SVD**, a training-free framework that optimizes across both column and module levels. Synergizing a *Column-Preserving Strategy* for sensitive weight columns and a *Module-Adaptive Allocation Strategy* for global budget management, Duo-SVD consistently outperforms SOTA baselines across LLaMA and Mistral families. Vali-

dated by superior reasoning performance, quantization compatibility, and inference speedups, Duo-SVD establishes a highly effective paradigm for efficient LLM deployment.

7 Limitations

Although Duo-SVD demonstrates significant reductions in perplexity and theoretical computational costs (FLOPs), a primary limitation lies in the lack of integration within high-performance inference frameworks such as vLLM. Specifically, the proposed *Column-Preserving Strategy* introduces a hybrid computation pattern involving both dense and low-rank pathways. Without specialized hardware-aware optimizations (e.g., fused CUDA kernels), the overhead of memory access and kernel launching in standard implementations may diminish the translation of theoretical compression ratios into real-world wall-clock speedups. Future work will focus on developing optimized kernels to fully realize the acceleration potential of Duo-SVD in production environments. Furthermore, regarding the theoretical analysis, the convexity of the objective function $\mathcal{F}(c)$ is derived based on an approximated upper bound of the reconstruction error. While extensive empirical observations consistently validate the unimodality of the exact loss landscape across diverse settings, a rigorous mathematical proof establishing the strict convexity of the exact $\mathcal{F}(c)$ remains an open theoretical question.

Despite the superior performance achieved by our proposed method, it introduces a non-negligible increase in pre-computation overhead, primarily due to the MAAS. As detailed in Table 4, for a LLaMA-2 13B model, the MAAS-based profiling requires approximately 12 hours on two RTX 3090 GPUs, whereas the baseline SVD-LLM completes its calibration in roughly 20 minutes.

Method	Calib. / MAAS	Decomp. / CPS	Total Time
SVD-LLM	∼20 min	∼20 min	∼40 min
Duo-SVD (Ours)	∼12 hours	∼40 min	∼12.7 hours

Table 4: Time cost breakdown for LLaMA-2 13B on two RTX 3090 GPUs.

However, we emphasize that model compression is typically a one-off offline process. The 12-hour profiling stage constitutes a singular investment that yields permanent performance benefits during the subsequent, prolonged inference phase. Given that the core decomposition step remains highly efficient (∼40 min), we argue that the substantial improvement in performance retention justifies this additional offline computational cost.

8 Ethical Considerations

This work promotes Green AI by reducing resource demands, thereby lowering the barriers to LLM deployment. However, it is important to acknowledge that low-rank decomposition and pruning techniques may inadvertently impact the model's safety alignment or amplify existing biases found in the pre-trained weights. Although our evaluations on standard benchmarks indicate robust performance retention, the potential for degradation in safety guardrails or fairness remains a critical consideration.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-Based Adaptive Structured Pruning for Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):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. *Preprint*, arXiv:2401.15024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439. Number: 05.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Merouane Debbah, and Deepak Gupta. 2024. Faster and lighter llms: A survey on current challenges and way forward. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 7980–7988. International Joint Conferences on Artificial Intelligence Organization.

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 [cs].

Carl Eckart and Gale Young. 1936. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika*, 1(3):211–218.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10323–10337. PMLR. ISSN: 2640-3498.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge Distillation of Large Language Models. *International Conference on Representation Learning*, 2024:32694–32717.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Preprint*, arXiv:2009.03300.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1404–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xinhao Huang, You-Liang Huang, and Zeyi Wen. 2025. SoLA: Leveraging Soft Activation Sparsity and Low-Rank Decomposition for Large Language Model Compression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(16):17494–17502.

Yixin Ji, Yang Xiang, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, Kehai Chen, and Min Zhang. 2024. Adaptive Feature-based Low-Rank Compression of Large Language Models via Bayesian Optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4152–4168, Miami, Florida, USA. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint*. ArXiv:2310.06825 [cs].

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2025. ModeGPT: Modular decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*.

LinJi, TangJiaming, TangHaotian, YangShang, XiaoGuangxuan, and HanSong. 2025. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *GetMobile: Mobile Computing and Communications*. Publisher: ACMPUB27New York, NY, USA.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. *Advances in Neural Information Processing Systems*, 36:21702–21720.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.

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.

Wang Qinsi, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. 2025. Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1):140:5485–140:5551.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. 2024. A Simple and Effective Pruning Approach for Large Language Models. *International Conference on Representation Learning*, 2024:4942–4964.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint*. ArXiv:2307.09288 [cs].

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. 2025a. Basis sharing: Cross-layer parameter sharing for large language model compression. In *The Thirteenth International Conference on Learning Representations*.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. 2025b. SVD-LLM V2: Optimizing Singular Value Truncation for Large Language Model Compression. *Preprint*, arXiv:2503.12340.

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

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*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 Technical Report. *Preprint*, arXiv:2505.09388.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. LaCo: Large Language Model Pruning via Layer Collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417, Miami, Florida, USA. Association for Computational Linguistics.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay 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 *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Hao Yu and Jianxin Wu. 2023. Compressing Transformers: Features Are Low-Rank, but Weights Are Not! *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11007–11015.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2024. ASVD: Activation-aware Singular Value Decomposition for Compressing Large Language Models. *arXiv preprint*. ArXiv:2312.05821 [cs].

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2025. BlockPruner: Fine-grained Pruning for Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5065–5080, Vienna, Austria. Association for Computational Linguistics.

Qihuang Zhong, Liang Ding, Li Shen, Juhua Liu, Bo Du, and Dacheng Tao. 2024. Revisiting Knowledge Distillation for Autoregressive Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10900–10913, Bangkok, Thailand. Association for Computational Linguistics.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A Survey on Efficient Inference for Large Language Models. *arXiv preprint*. ArXiv:2404.14294 [cs].

A Pseudocode for Duo-SVD

Algorithm 1 Duo-SVD Compression Framework

Require: Pre-trained Model $\mathcal{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_M\}$, Calibration Data $\mathcal{X}$, Global Target Rate R_{target}, Candidate Rate Set $\mathcal{R}$.
Ensure: Compressed Model $\mathcal{W}'$.

// Stage 1: Module-Adaptive Allocation
for each module $i \in \{1, \dots, M\}$ **do**
  for each rate $s \in \mathcal{R}$ **do**
    Compute sensitivity $E_{i,s}$ via Eq. (8) on $\mathcal{X}$.
  end for
end for
Solve optimal ratios $\mathbf{s}^* = [s^{(1)}, \dots, s^{(M)}]$ via Dynamic Programming subject to Eq. (9).
// Stage 2: Column-Preserving Compression
for each module $i \in \{1, \dots, M\}$ **do**
  if $s^{(i)} > 0$ **then**
    Compute column errors $\{L_{\text{col},j}\}_{j=1}^{n}$ via Eq. (4).
    Sort errors such that $L_{\text{col},(1)} \geq L_{\text{col},(2)} \geq \cdots \geq L_{\text{col},(n)}$.
    Determine $c^* = \arg\min_c \mathcal{F}(c)$ using **Ternary Search** based on the convexity of $\mathcal{F}(c)$.
    Set $\mathcal{S} \leftarrow \{(1), \dots, (c^*)\}$ and calculate rank $r(c^*)$.
    Decompose $\mathbf{W}_i$ into $\mathbf{W}_{\mathcal{S}}, \mathbf{W}_u, \mathbf{W}_v$ via Eq. (5).
    $\mathbf{W}'_i \leftarrow \{\mathbf{W}_{\mathcal{S}}, \mathbf{W}_u, \mathbf{W}_v\}$.
  else
    $\mathbf{W}'_i \leftarrow \mathbf{W}_i$
  end if
end for
return $\mathcal{W}' = \{\mathbf{W}'_1, \dots, \mathbf{W}'_M\}$

B Empirical Verification of Convexity

Following the preliminary analysis in Figure 3, we further empirically investigate the convexity of the reconstruction loss with respect to the column retention ratio. Figure 8 presents the loss landscapes across diverse settings, including different model layers, architectures, and compression ratios.

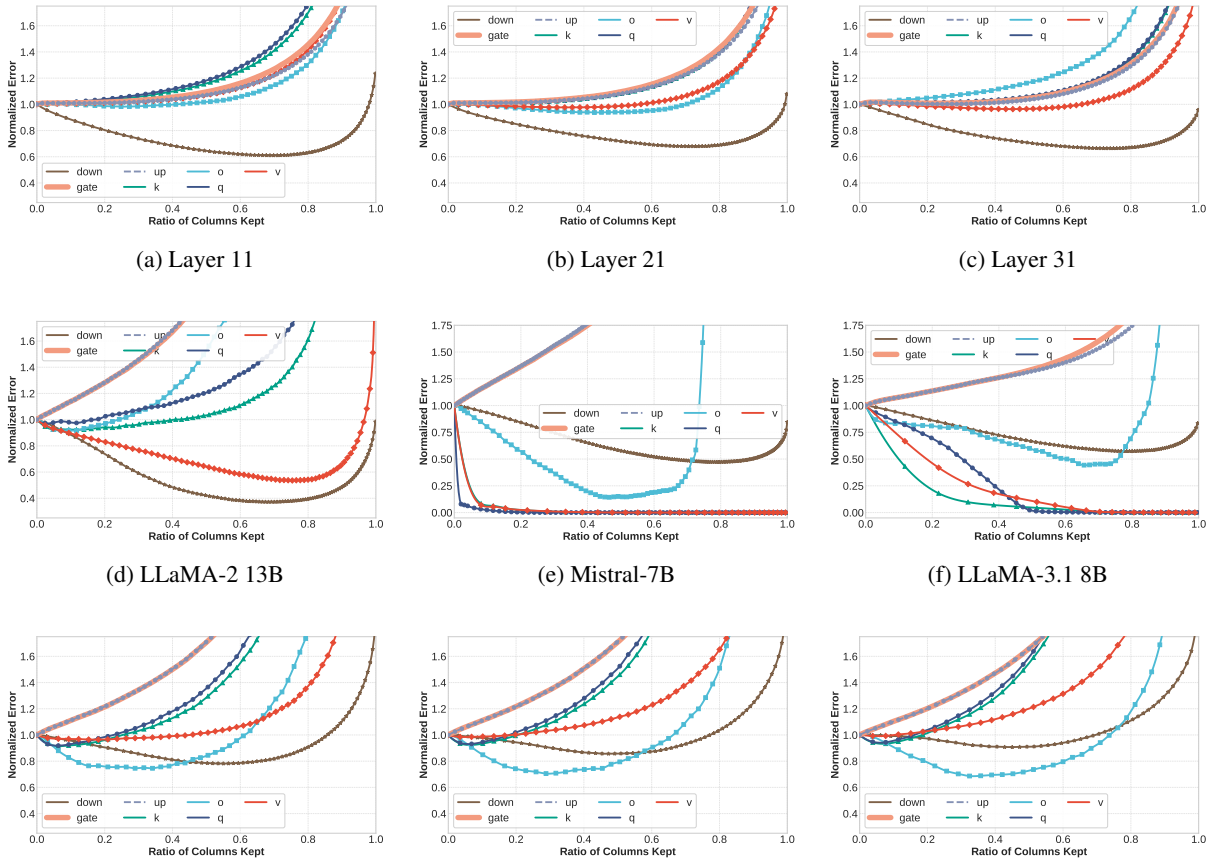
Observations indicate that while the specific loss magnitudes vary significantly across these settings, the curves consistently exhibit a unimodal characteristic—possessing a unique local minimum that coincides with the global minimum. These empirical findings strongly validate the approximation proposed in Eq. (7), providing a solid empirical basis for employing ternary search to efficiently determine the optimal number of retained columns.

C Derivation of Convexity for $m \geq n - c$

In the main text, we derived the gradient approximation under the assumption $m \leq n - c$. Here, we provide the derivation for the complementary case where $m \geq n - c$. Let $\mathbf{W}_c \in \mathbb{R}^{m \times (n-c)}$ and $\mathbf{W}_{c+k} \in \mathbb{R}^{m \times (n-c-k)}$. Since $m \geq n - c$, the summation of singular values extends only up to the number of columns. The discrete gradient is derived as follows:

$$
\begin{aligned}
&\Delta\mathcal{F}(c) \\
&= L^2_{\text{SVD},r(c)-1}(\mathbf{W}_{c+k}) - L^2_{\text{SVD},r(c)}(\mathbf{W}_c) \\
&= \sum_{i=r(c)}^{n-c-k} \sigma_i^2(\mathbf{W}_{c+k}) - \sum_{i=r(c)+1}^{n-c} \sigma_i^2(\mathbf{W}_c) \\
&= \sigma^2_{r(c)}(\mathbf{W}_{c+k}) \\
&\quad \underbrace{+ \sum_{i=r(c)+1}^{n-c-k} \left(\sigma_i^2(\mathbf{W}_{c+k}) - \sigma_i^2(\mathbf{W}_c)\right)}_{\text{Negative Term 1}} \\
&\quad \underbrace{- \sum_{i=n-c-k+1}^{n-c} \sigma_i^2(\mathbf{W}_c)}_{\text{Negative Term 2}} \\
&\approx \underbrace{\sigma^2_{r(c)}(\mathbf{W}_{c+k})}_{\leq \delta^2_{r(c)}(\mathbf{W})} - \sum_{j=1}^{k} L^2_{\text{col},(c+j)} \\
&\leq \delta^2_{r(c)}(\mathbf{W}) - \sum_{j=1}^{k} L^2_{\text{col},(c+j)}.
\end{aligned}
\tag{10}
$$

Similar to the main text, the *Negative Term* (consisting of the Cauchy interlacing difference and the tail singular values) represents the total loss of spectral energy due to removing k columns. We approximate this term using the loss of the newly preserved columns ($-\sum L^2_{\text{col}}$). Consequently, we arrive at the same inequality structure: the upper bound is composed of a increasing cost term $\delta^2_{r(c)}(\mathbf{W})$ and a decreasing gain term $\sum L^2_{\text{col}}$. Thus, the convexity of $\mathcal{F}(c)$ holds regardless of the matrix dimensions.



(a) Layer 11 (b) Layer 21 (c) Layer 31

(d) LLaMA-2 13B (e) Mistral-7B (f) LLaMA-3.1 8B

(g) 30% (h) 40% (i) 50%

Figure 8: Verification of the monotonicity of $\mathcal{F}(c)$. (a)–(c) Results for LLaMA-2 7B at a 20% compression ratio. (d)–(f) Results for Layer 0 at a 20% compression ratio. (g)–(i) Results for Layer 0 of LLaMA-2 7B.

Model	Method	WinoG.	HellaS.	ARC-e	ARC-c	PIQA	Avg. ↑
	Dense	69.06	75.99	74.58	46.25	77.91	68.76
LLaMA-2 7B	LaCo	60.46	54.08	55.39	35.84	68.34	54.82
	ShortGPT	65.90	62.63	56.06	36.09	70.24	58.18
	BlockPruner	62.43	65.87	61.07	37.29	74.21	60.17
	Basis Sharing	**66.54**	56.22	59.18	30.82	66.54	55.86
	Dobi-SVD	58.56	56.80	54.25	26.29	65.34	52.25
	Duo-SVD	65.67	**70.23**	**70.50**	**40.53**	**76.77**	**64.74**

Table 5: Performance comparison of different methods on LLaMA-2 7B at 20% compression ratio

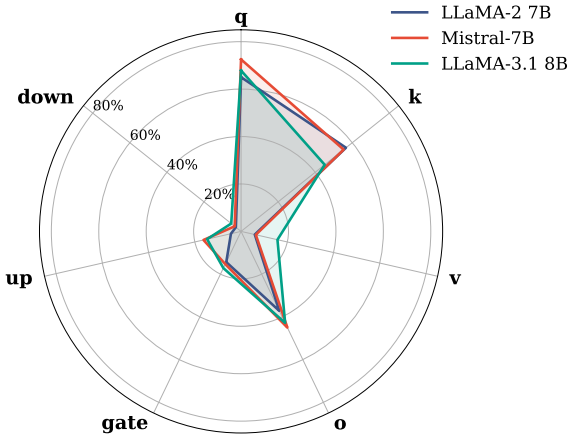
Ratio	Method	PPL ↓	Avg. ↑	MMLU-5shot	PIQA	WinoG.	HellaS.	ARC-e	ARC-c	OBQA
Dense	–	5.11	62.14	45.70	79.05	69.38	75.92	74.49	46.25	44.20
20%	BayesOpt	6.50	55.24	33.84	73.99	**65.98**	64.08	69.19	37.97	41.60
	TEM	8.05	46.26	28.68	70.95	63.77	55.98	60.48	34.13	38.80
	MAAS	**5.99**	**57.27**	**35.41**	**76.77**	65.67	**70.23**	**70.50**	**40.53**	**41.80**
30%	BayesOpt	7.85	50.45	**29.87**	70.18	**65.11**	55.87	59.97	33.36	**38.80**
	TEM	11.60	44.12	29.68	63.28	60.22	44.28	49.07	28.50	33.80
	MAAS	**7.50**	**51.11**	27.09	**72.14**	63.69	**60.97**	**61.28**	**34.13**	**38.80**

Table 6: Comparison of WikiText-2 perplexity and downstream task accuracy of different methods on LLaMA-2 7B.

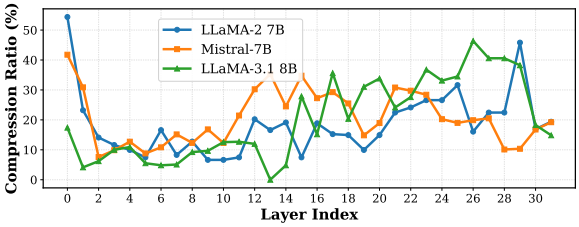
D Additional Comparison of Different Methods

In this section, we present a comparative analysis of Duo-SVD against an expanded set of baselines on zero-shot tasks. Specifically, LaCo, ShortGPT, and BlockPruner represent depth pruning approaches that operate at the granularity of Transformer layers or individual Attention and MLP modules. In contrast, Basis Sharing and Dobi-SVD are founded on SVD decomposition. To ensure a fair comparison, we evaluate the unquantized version of Dobi-SVD. As reported in Table 5, Duo-SVD outperforms the baseline methods by a margin of at least 4 percentage points in average accuracy, demonstrating superior performance retention capabilities.

E Additional Results of MAAS



(a) Average compression ratio across different module types



(b) Per-layer compression ratios

Figure 9: Allocation results of different models at 20% compression ratio

In this section, we present the MAAS results regarding the compression rates of distinct module types and layers for LLaMA-2 7B, Mistral-7B, and LLaMA-3.1 8B, all evaluated at a 20% compression ratio. As illustrated in Figure 9, the models exhibit consistent patterns: the compression ratios for Attention modules are significantly higher than those for MLP modules, and the latter layers are compressed more heavily than the earlier ones. However, distinct model-specific behaviors are also observable. Specifically, in LLaMA-3.1 8B, the V modules undergo more aggressive compression, whereas the K modules retain a larger proportion of parameters. In contrast, Mistral-7B exhibits relatively higher compression rates in its intermediate layers. Collectively, these findings provide novel data for analyzing the relative importance of specific modules and layers across different model architectures.

F Comparison of Different Allocation Strategy

To evaluate the effectiveness of MAAS, we compare it against distinct compression ratio allocation strategies. For clarity in Table 6, we denote the Bayesian optimization framework employed by Bolaco (Ji et al., 2024) as BayesOpt, and the cumulative truncation error minimization approach adopted by SoLA (Huang et al., 2025) as TEM. To ensure a fair comparison, we uniformly employ CPS as the underlying decomposition technique across all allocation methods.

The results for LLaMA2-7B at 20% and 30% compression ratios, presented in Table 6, demonstrate that MAAS consistently achieves superior performance in both perplexity and downstream task accuracy. The primary advantage of MAAS stems from its capability to directly quantify the contribution of each module to the model's overall performance. Unlike BayesOpt, which necessitates coarse-grained rank sharing to handle optimization dimensionality, or TEM, which relies on truncation error as an indirect proxy, MAAS enables a fine-grained and precise rank allocation. This direct evaluation mechanism allows MAAS to effectively preserve critical information, thereby yielding the significant performance gains observed over the baselines.

G Theoretical Analysis of Efficiency

Based on the dual-pathway formulation defined in Eq. 5 and the optimization constraint in Eq. 6, we analyze the theoretical efficiency of Duo-SVD.

Storage Efficiency. The storage efficiency is directly enforced by the optimization constraint. For a target compression ratio $\rho \in (0, 1)$, the total parameter budget is strictly bounded by $N_{\text{Duo}} \leq \rho mn$. Assuming the budget is fully utilized, the storage reduction ratio is trivially $N_{\text{Duo}}/N_{\text{orig}} = \rho$.

Computational Efficiency. To prove the linear reduction in computation, we express the total parameter count N_{Duo} as the sum of the dense and low-rank components:

$$N_{\text{Duo}} = mc + r(m + n - c) = \rho mn. \quad (11)$$

The total computational cost $\mathcal{C}_{\text{Duo}}$ (in FLOPs) for a batch size d aggregates the operations from both paths:

$$\mathcal{C}_{\text{Duo}} = \underbrace{d \cdot mc}_{\text{Dense Path}} + \underbrace{d \cdot r(n - c) + d \cdot mr}_{\text{Low-Rank Path}}. \quad (12)$$

By factoring out d and rearranging the terms, we observe that the bracketed expression is identical to the parameter definition in Eq. 11:

$$\begin{aligned} \mathcal{C}_{\text{Duo}} &= d \cdot [mc + r(m + n - c)] \\ &= d \cdot N_{\text{Duo}} \\ &= d \cdot \rho mn. \end{aligned} \quad (13)$$

Since the original computational cost is $\mathcal{C}_{\text{orig}} = d \cdot mn$, the reduction ratio is exactly $\mathcal{C}_{\text{Duo}}/\mathcal{C}_{\text{orig}} = \rho$. This confirms that both storage and computation scale linearly with the compression ratio ρ.