DipSVD: Dual-importance Protected SVD for Efficient LLM Compression

Anonymous ACL submission

Abstract

The ever-increasing computational demands and deployment costs of large language models (LLMs) have spurred numerous compressing methods. Compared to quantization and unstructured pruning, SVD compression offers superior hardware compatibility and theoretical guarantees. However, existing SVD-based methods focus on the overall discrepancy between the original and compressed matrices while overlooking the protection of critical components within the matrix, which leads to inferior performance in the compressed models. This paper proposes a dual-level importance protection mechanism to enhance SVD-based compression methods: (1) local importance protection: preserving the most critical singular vectors within each weight matrix through channel-weighted data whitening; and (2) global importance protection: enabling less important layers to bear a greater portion of the compression burden through either a heuristic or optimization-based approach, thereby minimizing the impact of compression on critical layers. Extensive experiments demonstrate that DipSVD outperforms existing SVD-based compression approaches across multiple benchmarks, achieving superior model performance especially at high model compression ratios.

1 Introduction

While Large Language Models (LLMs) demonstrate remarkable capabilities across diverse natural language tasks such as multi-round conversation (Chen et al., 2023; Long, 2023) and logical reasoning (Creswell et al., 2022; Duan et al., 2024; Pan et al., 2023), the ever-increasing model scales impose severe computational burdens (Zhou et al., 2024; Wang et al., 2024a). This has spurred intensive research into LLM-specific compression techniques, including quantization (Frantar et al., 2022; Lin et al., 2024), parameter pruning (Men et al., 2024; Ma et al., 2023; Kim et al., 2024; Song et al., 2024; Ding et al., 2025), and knowledge distillation (Gu et al., 2023; Hsieh et al., 2023). While quantization and pruning require specialized hardware support and costly retraining, low-rank decomposition methods like Singular Value Decomposition (SVD) offer hardware-agnostic compression through dense matrix operations. Moreover, the KV cache of LLMs compressed via SVD at runtime can also be reduced (Wang et al., 2024b).

Method	Local Importance	Global Importance	Coupling Modeling
FWSVD	✓ Row-wise Fisher weighting	X	X
ASVD	X	✓ Sensity-based Truncation Rank Searching	X
SVD-LLM	X	X	X
Ours	✓ Channel-weighted Data Whitening	✓ Layer-Specific Compression	✓ Cross-Hierarchy Joint Optimization

Table 1: Comparison of existing SVD-based methods.

Despite these advantages, existing SVD-based compression methods are undermined by their failure to holistically consider both **local importance** (e.g., intra-layer channel sensitivity) and **global importance** (e.g., layer-wise heterogeneity) during matrix factorization (as shown in Tab.1): (1) Globally-aware methods like ASVD (Yuan et al., 2023) dynamically allocate compression ratios across layers but retain standard SVD decomposition within each layer, risking excessive pruning of sensitive local features. (2) Locally-aware methods such as FWSVD (Hsu et al., 2022) weight intra-layer channels but ignore global disparities. In LLMs, where layers with same structure have different roles (Zhang et al., 2024), uniform compression across these heterogeneous layers leads to suboptimal efficiency. (3) Isotropic methods including SVD-LLM (Wang et al., 2024b) apply homogeneous compression without importance weighting at either level and inevitably degrade the performance-efficiency trade-offs. Critically, no existing work jointly optimizes global and local importance during SVD decomposition—a gap we empirically prove to be detrimental under aggressive compression.

In this paper, we propose DipSVD, a **D**ual-

importance **p**rotected SVD-based compression method. Specifically, our approach introduces: (1) Local Importance Protection: By employing channel-weighted data whitening, this method preserves the most importance singular vectors in the weight matrix, while allowing less important singular vectors to bear a greater portion of the compression burden. (2) Global Importance Protection: Through either an optimization-based approach or a heuristic method derived from layer-wise gradient sensitivity analysis, we automatically determine the optimal layer-specific compression ratios, thereby protecting the most critical layers of the model and allocating more compression burden to less importance layers. Through Pearson correlation analysis, we verify that the heuristic compression ratios closely align with those obtained from Bayesian optimization, while simultaneously reducing computation overhead.

We conduct extensive experiments to evaluate the effectiveness of our DipSVD method, benchmarking against three SVD-based compression methods across five LLMs of varying architectures and scales. Our evaluation encompasses both zero-shot task performance and perplexity metrics under identical experimental conditions. Experimental results demonstrate the superiority of DipSVD in terms of both zero-shot task performance and generation quality. Additionally, our ablation studies show that with one of the two key components of DipSVD alone, it still outperforms state-of-the-art SVD-based compression methods under different compression ratios.

The contributions of this study are summarized as:

- We propose DipSVD, a novel compression framework for LLMs that explicitly integrates both local and global importance into the SVD process, aiming to preserve model integrity and task performance.

- To implement this framework, we (i) introduce an importance-aware whitening mechanism to efficiently estimate compression loss while emphasizing local importance, and (ii) develop two strategies for global importance modeling: a high-performance Bayesian optimization method and a lightweight heuristic-based alternative.

- Extensive experiments demonstrate that DipSVD effectively improves on-device inference efficiency while maintaining superior performance across seven diverse zero-shot tasks and perplexity benchmarks.

2 Related Work

2.1 Large Language Model Compression

To mitigate the computational and memory demands of LLMs, researchers have proposed multiple compression techniques such as quantization, unstructured pruning, structured pruning and knowledge distillation. Although these methods have demonstrated practical effectiveness, each has its own limitations. Quantized methods like GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2024), and SmoothQuant (Xiao et al., 2023) enable low-bit inference but often sacrifice accuracy or hardware efficiency. Unstructured pruning methods like SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2023) and GBLM-Pruner (Das et al., 2023) can remove 50–75% of weights with reasonable accuracy, though their practical speedups on general-purpose hardware remain limited. While structured pruning methods (Chen et al., 2021; Ma et al., 2023; An et al., 2024) achieve hardware acceleration through removing entire architectural components, they suffer from substantial accuracy loss under aggressive pruning. Recent knowledge distillation methods such as GKD (Agarwal et al., 2023) and DistiLLM (Ko et al., 2024) compress auto-regressive models into smaller students, but require substantial computation and retraining. Unlike these methods, low-rank approximation based on Singular Value Decomposition (SVD) offers an efficient alternative, requiring no retraining while maintaining hardware compatibility.

2.2 Low-rank Decomposition

SVD compression is a widely used technique to reduce matrix size by decomposing the weight matrix and truncating smaller singular values (Kim et al., 2015). Specifically, FWSVD (Hsu et al., 2022) incorporates Fisher information to prioritize important parameters, better maintaining prediction accuracy. ASVD (Yuan et al., 2023) scales the weight matrix based on activation distributions and adjusts the compression ratio layer-wise, allowing performance preservation at moderate compression ratios without retraining. SVD-LLM (Wang et al., 2024b) introduces truncation-aware data whitening and closed-form layer-wise parameter updates,

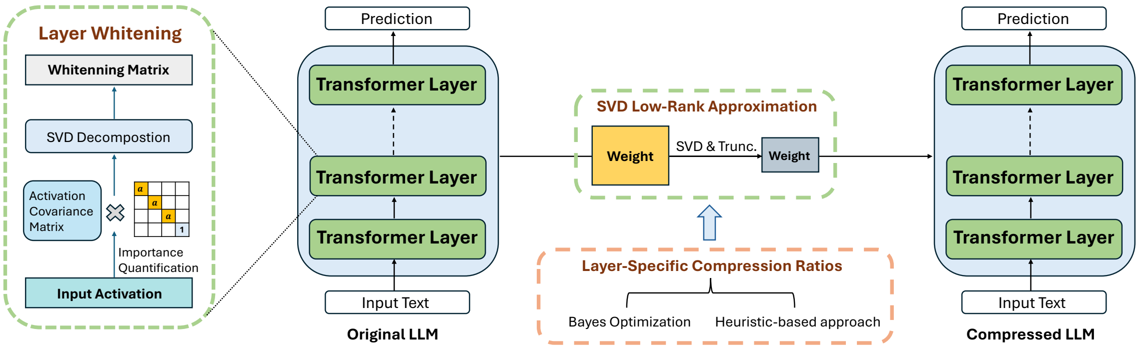

Figure 1: Overview of DipSVD.

significantly improving the balance between compression efficacy and inference speed.

However, existing SVD-based methods typically neglect both global and local importance considerations in model architecture. The failure to account for global importance, as evidenced by varying layer-wise compression sensitivity, leads to suboptimal rank selection and significant performance degradation under high compression ratios (Ding et al., 2025). Concurrently, ignoring local importance, manifested through unequal channel contributions to model outputs, may inadvertently prune structurally critical channels and cause substantial accuracy deterioration.

3 Method

Fig.1 provides an overview of DipSVD. Following the standard procedure of post-training LLM compression methods, DipSVD first uses a random set of sentences as calibration data to generate activation for local importance protection and layer whitening. Specifically, DipSVD selectively emphasize important channels in the whitening transformation, which not only ensures a direct mapping between singular values and compression loss, but also improves structural retention during compression. To preserve global importance of LLMs, DipSVD groups the weight matrices across by Transformer layer in the original LLM. For each Transformer layer, DipSVD assigns a unique compression ratio which computed through Bayes Optimization or Heuristic-based approach. Lastly, DipSVD applies the layer-specific compression ratios to the weight matrix and performs SVD to truncate the weight matrices to compress the LLM. The following subsections provide a detailed description of each protection method and their synergistic integration. Pseudocode is provided in Appendix.A.

3.1 Local Importance Protection

3.1.1 Channel-weighted Whitening

To preserve structurally important channels during whitening, we propose an importance-aware transformation that adapts to the second-order statistics of the input. Let $X \in \mathbb{R}^{m \times n}$ denote a data matrix with m samples and n feature channels.

We assess the structural importance of each channel by evaluating how much it contributes to the overall sample-level second-order structure. Specifically, for the j-th feature channel $x_j = X_{:,j} \in \mathbb{R}^m$, we define its importance as:

$$\alpha_j = \sqrt{x_j^T (XX^T) x_j}. \tag{1}$$

Channels with larger α_j values are considered more significant, as they exert a stronger influence on the global sample structure encoded in XX^T. Geometrically, this expression can be viewed as the magnitude of the projection of channel vector x_j onto the principal subspace spanned by the data samples. In other words, α_j^2 reflects how well the direction x_j aligns with the dominant variance structure in the sample space. Channels with high alignment are thus deemed structurally important and are prioritized for preservation in the subsequent whitening step.

To enhance the preservation of such channels during whitening, we introduce a diagonal scaling matrix $D \in \mathbb{R}^{n \times n}$, defined by:

$$D_{jj} = \begin{cases} a, & \text{if } \alpha_j \text{ is among the top } p\% \text{ values,} \\ 1, & \text{otherwise,} \end{cases} \tag{2}$$

with $a > 1$. This results in a reweighted input $\tilde{X} = XD$, where structurally important channels are selectively amplified.

As a preparatory step for whitening, we compute the second-order structure of $\tilde{X}$ through the matrix product:

$$\tilde{X}^T\tilde{X} = D^T X^T X D. \tag{3}$$

Applying singular value decomposition, we express this matrix as $\tilde{X}^T\tilde{X} = U_{\tilde{X}}\Sigma_{\tilde{X}}U_{\tilde{X}}^T$, where $U_{\tilde{X}} \in \mathbb{R}^{n\times n}$ is orthogonal and $\Sigma_{\tilde{X}}$ is diagonal with non-negative entries. We then construct the whitening matrix:

$$S = \Sigma_{\tilde{X}}^{-1/2}U_{\tilde{X}}^T, \tag{4}$$

which yields the final whitened output as $\hat{X} = \tilde{X}S = XDS$. This process results in decorrelated, variance-normalized features, while maintaining the structural contributions of high-importance channels identified through the original covariance.

3.1.2 Impact of Whitening on Truncation

The channel-weighted whitening operation amplifies the contributions of important channels while ensuring that the compression loss is directly related to the singular values of the weight matrix, which is critical for minimizing the performance degradation caused by compression. In the following, we provide a theoretical derivation explaining why the whitening process guarantees a direct mapping between singular values and compression loss.

Singular Value Decomposition of Whitened Weight Matrix. We first perform SVD on the whitened weight matrix WS to obtain its decomposition:

$$WS = U\Sigma V^T = \sum_{i=1}^{r}\sigma_i u_i v_i^T, \tag{5}$$

where: $U = [u_1, u_2, ..., u_r]$ contains the left singular vectors $u_i \in \mathbb{R}^n$, $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_r)$ contains the singular values in descending order, $V = [v_1, v_2, ..., v_r]$ contains the right singular vectors $v_i \in \mathbb{R}^m$, r is the rank of WS.

Single Singular Value Truncation. When truncating the i-th singular value σ_i of WS, the compression loss L_i is given by:

$$\begin{aligned} L_i &= \|W\tilde{X} - W'\tilde{X}\|_F \\ &= \|(WS - \text{SVD}(WS))S^{-1}\tilde{X}\|_F \\ &= \|\sigma_i u_i v_i^T S^{-1}\tilde{X}\|_F. \end{aligned} \tag{6}$$

Leveraging the orthonormality of u_i and v_i (i.e., $u_i^T u_i = v_i^T v_i = 1$), the whitening property $S^{-1}\tilde{X}\tilde{X}^T(S^{-1})^T = I$, and the invariance of the Frobenius norm under orthogonal transformations, we obtain:

$$\begin{aligned} L_i &= \sigma_i\left[\text{trace}\left(v_i^T S^{-1}\tilde{X}\tilde{X}^T(S^{-1})^T v_i\right)\right]^{1/2} \\ &= \sigma_i\left[v_i^T \cdot I \cdot v_i\right]^{1/2} \\ &= \sigma_i. \end{aligned} \tag{7}$$

This shows that truncating a single singular value results in a compression loss equal to that singular value.

Multiple Singular Value Truncation. When truncating the smallest $r - m$ singular values $\{\sigma_{m+1}, \ldots, \sigma_r\}$ of WS, the total compression loss is defined as the output difference on input $\tilde{X}$:

$$L = \left\|\sum_{i=m+1}^{r}\sigma_i u_i v_i^T S^{-1}\tilde{X}\right\|_F = \sqrt{\sum_{i=m+1}^{r}\sigma_i^2}. \tag{8}$$

In summary, truncating smaller singular values minimizes both the compression loss and approximation error, where the total impact is determined by the root-sum-square of the truncated singular values. The whitening operation ensures this direct correspondence between singular values and model performance.

3.2 Global Importance Protection

While local importance protection effectively preserves critical channels within individual layers, a key challenge lies in ensuring that globally important layers are also adequately protected. To this end, we propose two compression strategies based on global layer-wise importance, which allocate different compression ratios to different layers according to their relative significance.

3.2.1 Bayesian Optimization for Layer-Specific Compression

For scenarios where computational resources are abundant and the highest possible performance is desired, Bayesian optimization can be employed to directly optimize the global compression objective. This approach searches for the optimal compression ratios by maximizing the cosine similarity between the outputs of the original and compressed models, subject to the global compression constraint:

$$\begin{aligned} &\max_{k_1,k_2,\ldots,k_L} \text{cosine\_similarity}\big(f_{\text{orig}}(x), f_{\text{comp}}(x)\big) \\ &\qquad\text{s.t. } \frac{1}{L}\sum_{l=1}^{L}k_l = k, \end{aligned} \tag{9}$$

where k_l are layer compression ratios, L is total layers, and k is the target global compression ratio.

3.2.2 Efficient Heuristic for Layer-Specific Compression

While Bayesian optimization offers superior performance, it is computationally expensive and may not be necessary for all applications. A practical and efficient alternative is provided by a heuristic-based approach that combines two key metrics: Fisher Sensitivity and Effective Rank. These metrics provide complementary insights into the importance and compressibility of each layer, enabling a cost-effective yet principled compression strategy.

Fisher Sensitivity. Fisher Sensitivity measures how sensitive the model's loss is to changes in the parameters of each layer. It is computed as the ratio of the gradient norm to the parameter norm for each layer, capturing the relative importance of the layer's parameters. Specifically, for each layer l, Fisher Sensitivity can be formulated as:

$$S_l = \sum_{\text{Attention}} \frac{\|\nabla_\theta \mathcal{L}\|_F}{\|\theta\|_F} + \sum_{\text{MLP}} \frac{\|\nabla_\theta \mathcal{L}\|_F}{\|\theta\|_F}, \qquad (10)$$

where $\nabla_\theta \mathcal{L}$ is the gradient of the loss with respect to the parameters θ and $\|\cdot\|_F$ denotes the Frobenius norm.

Effective Rank. Effective Rank quantifies the information density of each layer's output by analyzing its singular value distribution. For each layer l, we compute the singular values of its output matrix $H_l \in \mathbb{R}^{B\times T\times D}$ (where B is the batch size, T is the sequence length, and D is the hidden dimension) and determine the smallest rank R_l that captures a predefined threshold (e.g., 95%) of the cumulative energy:

$$R_l = \min\left\{ k \mid \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{r} \sigma_i} \geq \text{threshold} \right\}, \qquad (11)$$

where σ_i are the singular values of H_l.

Combining Sensitivity and Effective Rank. Layers with higher sensitivity values are more critical to the model's performance and should be preserved more aggressively. Similarly, layers with lower effective ranks are more compressible and can tolerate higher compression. To assign compression ratios to each layer, we first combine the Fisher Sensitivity S_l and Effective Rank R_l into a unified importance score Q_l:

$$Q_l = (S_l)^\beta \cdot (R_l)^{1-\beta}, \qquad (12)$$

where β is a hyperparameter that controls the relative importance of sensitivity and effective rank.

Given normalized importance scores Q_l for each layer and a target global compression ratio k, we define the per-layer preservation ratios p_l (i.e., the proportion of parameters retained in each layer) such that the average preservation ratio across all layers equals $1 - k$. The preservation ratios are computed as:

$$p_l = \frac{Q_l}{\sum_{j=1}^{L} Q_j} \cdot L \cdot (1 - k). \qquad (13)$$

The corresponding compression ratios are then given by $1 - p_l$. This formulation ensures that layers with higher importance scores Q_l are compressed less aggressively, while maintaining the desired global compression budget. As shown in Fig.2, by capturing both the sensitivity trends S_l and compressibility patterns R_l, the derived preservation ratios clearly reflect each layer's relative contribution to model performance.

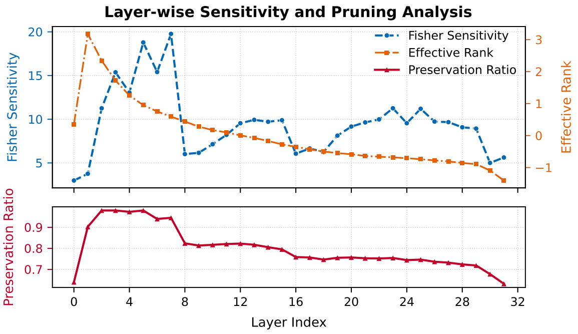

Figure 2: The Q_l, S_l, and R_l values for each layer in Vicuna-7B at 20% compression ratio.

3.3 Integrated Compression Process

DipSVD incorporates local and global significance protection mechanisms into compression:

Step 1: Layer Whitening. For each layer, perform data whitening using the matrix $\tilde{X}^T\tilde{X}$ and SVD, as described in Sec.3.1.1.

Step 2: Layer-Specific Compression Ratios. Use Bayesian optimization (or the heuristic-based approach) to determine the optimal compression ratios $k_1, k_2, \ldots, k_L$ for each layer, as described in Sec.3.2.

Step 3: SVD Low-Rank Approximation. For each layer l, apply the compression ratio k_l to the weight matrix W_l using SVD-based low-rank approximation:

1) Perform SVD on the whitened weight matrix $W_l S$:

$$W_l S = U_l \Sigma_l V_l^T, \qquad (12)$$

Model	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
LLaMA-7B	ASVD	95.268	200.937	86.269	0.186	0.379	0.557	0.333	0.242	0.607	0.218	0.360
	FWSVD	33.001	53.587	38.240	0.186	0.507	0.572	0.343	0.242	0.632	0.217	0.386
	SVD-LLM	9.526	28.967	26.390	0.242	0.509	0.570	0.352	0.269	0.630	0.227	0.400
	Ours	**9.427**	**22.270**	**19.909**	**0.242**	**0.602**	**0.640**	**0.405**	**0.296**	**0.661**	**0.230**	**0.440**
Vicuna-7B	ASVD	91.388	415.615	136.157	0.158	0.335	0.503	0.287	0.208	0.556	0.205	0.322
	FWSVD	43.690	239.318	64.753	0.172	0.459	0.545	0.312	0.224	0.613	0.221	0.364
	SVD-LLM	12.416	124.506	39.528	0.244	0.506	0.570	0.353	0.270	0.629	0.228	0.400
	Ours	**12.144**	**81.089**	**28.837**	**0.248**	**0.573**	**0.597**	**0.384**	**0.293**	**0.659**	**0.232**	**0.427**
DeepSeek-7B	ASVD	85.169	87.709	79.853	0.154	0.390	0.516	0.312	0.213	0.610	0.210	0.344
	FWSVD	68.416	99.775	118.319	0.142	0.406	0.551	0.296	0.194	0.595	0.220	0.344
	SVD-LLM	10.841	30.747	32.622	0.260	0.589	0.609	0.384	0.283	0.670	0.232	0.432
	Ours	**9.895**	**20.977**	**22.558**	**0.276**	**0.628**	**0.631**	**0.415**	**0.312**	**0.700**	**0.239**	**0.457**

Table 2: Performance of LLaMA-7B, Vicuna-7B and Deepseek-7B models compressed by DipSVD and baselines at 30% compression ratio, evaluated on three language modeling datasets (measured by perplexity) and seven classification datasets (measured by average accuracy). The best performance for each case is marked in bold.

where $U_l \in \mathbb{R}^{n\times r}$ and $V_l \in \mathbb{R}^{m\times r}$ are orthogonal matrices, and $\Sigma_l \in \mathbb{R}^{r\times r}$ is a diagonal matrix containing the singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$ of $W_l S$.

2) Truncate the smallest singular values in Σ_l based on k_l, obtaining the truncated diagonal matrix $\text{Trunc}_*(\Sigma_l)$.

3) Construct the compressed weight matrix W_l':

$$W_l' = U_l \times \text{Trunc}_*(\Sigma_l) \times V_l^T \times S^{-1}. \tag{13}$$

4) To further reduce memory usage, replace the original weight matrix W_l with two low-rank matrices $W_{u,l} \in \mathbb{R}^{n\times\tilde{r}}$ and $W_{v,l} \in \mathbb{R}^{\tilde{r}\times m}$:

$$W_{u,l} = U_l \times [\text{Trunc}_*(\Sigma_l)]^{1/2}, \tag{14}$$

$$W_{v,l} = [\text{Trunc}_*(\Sigma_l)]^{1/2} \times V_l^T \times S^{-1}, \tag{15}$$

where $\tilde{r}$ is the rank after truncation.

4 Experiments

4.1 Experimental setup

Foundation LLMs. We conducted experiments on existing popular LLMs at various scales, including LLaMA-{7B, 13B} (Touvron et al., 2023), Vicuna-{7B, 13B}-v1.5 (Chiang et al., 2023) and Deepseek-7B (DeepSeek-AI et al., 2025).

Baselines. We compared DipSVD with several previous SVD-based compression methods including FWSVD (Hsu et al., 2022), ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2024b).

Benchmarks. We measure zero-shot accuracy on commonsense reasoning datasets (i.e., PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy (Clark et al., 2018), ARC-challenge (Clark et al., 2018), MathQA(Amini et al., 2019) and OpenbookQA (Mihaylov et al., 2018a)) using the lm-evaluation-harness package (Gao et al., 2024). To assess sequence prediction performance, we report perplexity for DipSVD and the baselines on WikiText-2 (Merity et al., 2016), PTB (Marcus et al., 1993) and C4 (Mihaylov et al., 2018b).

Implementation Details. To ensure a fair comparison, we followed ASVD (Yuan et al., 2023) to randomly select 256 samples from WikiText-2 as the calibration data. All of our experiments are conducted on NVIDIA A100 GPUs.

4.2 Overall Performance

We evaluate the overall performance of DipSVD from three aspects: (1) performance on different LLMs, (2) performance on LLMs with larger scales, (3) performance under different compression ratios. Detailed results, contents generated by the compressed LLMs, and an analysis of the computational gains achieved by our method are provided in the Appendix.

Performance on different LLMs. We first compare the zero-shot task performance and perplexity metrics between DipSVD and the baseline on three different LLMs, including LLaMA-7B, Vicuna-7B and DeepSeek-7B under 30% compression ratio. As shown in Tab.2, DipSVD consistently outperforms the baselines across all three LLMs and all ten datasets. Specifically, we achieve the lowest perplexity (9.427) on LLaMA-7B model. For downstream tasks, our method attains the highest average accuracy (0.457 vs. SVD-LLM's 0.432 on DeepSeek-7B), with notable gains on reasoning-heavy benchmarks like ARC-Challenge (+4.3%).

These results collectively show that DipSVD effectively reduces model size and complexity while better preserving model performance compared to existing methods. The consistent improvements across different model architectures further demonstrate the robustness and generalizability of our proposed method.

Model	Method	WikiText-2	PTB	C4	Average
LLaMA-13B	ASVD	17.648	32.963	20.866	0.425
	FWSVD	12.963	22.123	18.509	0.383
	SVD-LLM	**7.618**	17.823	18.825	0.449
	Ours	7.697	**15.681**	**14.614**	**0.472**
Vicuna-13B	ASVD	28.309	637.196	39.799	0.401
	FWSVD	32.715	310.304	47.408	0.383
	SVD-LLM	9.616	145.715	29.204	0.449
	Ours	**9.070**	**49.948**	**19.203**	**0.458**

Table 3: Perplexity on three language modeling datasets and average accuracy of seven datasets of LLaMA-13B and Vicuna-13B at 30% compression ratio.

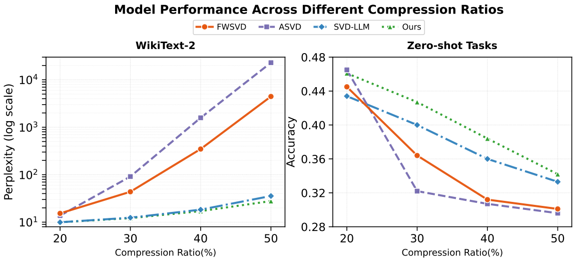

Figure 3: Perplexity on WikiText-2 and average zero-shot accuracy of Vicuna-7B compressed by DipSVD and baselines under 20% to 50% compression ratios.

Performance on LLMs with larger scales. We compare the performance between DipSVD and the baselines on LLaMA-13B and Vicuna-13B under 30% compression ratio on ten datasets in Tab.3. Specifically, our DipSVD method reduce PTB perplexity by 65.7% compared to SVD-LLM (49.948 vs 145.715) for Vicuna-13B. The results demonstrate that our method effectively scales to LLMs while maintaining superior capability preservation compared to existing compression methods.

Performance under Different Compression Ratios. We compare the performance between DipSVD and the baselines on Vicuna-7B under compression ratio ranging from 20% to 50% on WikiText-2 datasets and seven classification datasets. As shown in Fig.3, DipSVD consistently outperforms all baselines, and the performance gap gain with the compression ratio increases.

4.3 Performance of heuristics method

The heuristic method also serves as a key strategy for global importance preservation. Tab.4 compares perplexity scores across SVD-LLM, Bayesian optimization method, and the heuristic method. The results demonstrate that the heuristic method (DipSVD_H) significantly outperforms SVD-LLM across multiple compression ratios on all three datasets. While the Beyesian optimization method (DipSVD_B) achieves the best performance with large computational overheads, DipSVD_H remains competitive while offering a practical trade-off between performance and efficiency.

Compression Ratio	Method	WikiText2	PTB	C4
0.2	SVD-LLM	**9.942**	71.366	23.358
	DipSVD_B	9.952	56.869	**19.722**
	DipSVD_H	9.988	**54.380**	19.950
0.3	SVD-LLM	12.416	124.506	39.528
	DipSVD_B	**12.144**	**81.089**	**28.837**
	DipSVD_H	12.378	82.872	30.748
0.4	SVD-LLM	18.346	261.100	77.706
	DipSVD_B	**17.085**	**142.752**	**49.183**
	DipSVD_H	17.290	168.404	54.794
0.5	SVD-LLM	35.569	615.591	185.780
	DipSVD_B	**27.807**	**375.093**	**111.996**
	DipSVD_H	30.180	390.512	118.873

Table 4: Heuristics result of compressed Vicuna-7B. DipSVD_B represents compression model by Bayesian optimization method. DipSVD_H represents compression model by the heuristic method.

Connection with Bayesian optimization. In the experiments, the heuristic method showed the strongest correlation with the optimization-based method when $\beta = 0.25$. Therefore, we uniformly set β to 0.25. Tab.5 shows the Pearson correlation coefficient between the layer-wise compression rates obtained by heuristic and Bayesian optimization across different target compression ratios. The correlation coefficients consistently exceed 0.64, indicating strong agreement between the two methods at all compression levels.

Compression Ratio	0.2	0.3	0.4	0.5
Pearson Correlation Coefficient	0.645	0.669	0.711	0.706

Table 5: Pearson correlation coefficients between heuristic and Bayesian optimization methods across different compression ratios.

4.4 Ablation Study

4.4.1 Hyperparametric Ablation Experiment

We conducted ablation experiments to evaluate the impact of two hyperparameters in DipSVD: weight (importance amplification factor) and bar (top channel selection ratio). The results are shown in Fig.4. First, we isolated the effect of weight by testing progressive values from 0 to 150 while fixing bar=0.03, revealing that higher weights introduce better performance as the parameters increase until it remains unchanged. Next, we analyzed bar with values from 0 to 0.3, demonstrating that the performance will be better and then worse as the parameters increase. The optimal performance of both DipSVD (Step1) and DipSVD (Step1+Step2) can be achieved when weight=30 and bar=0.03.

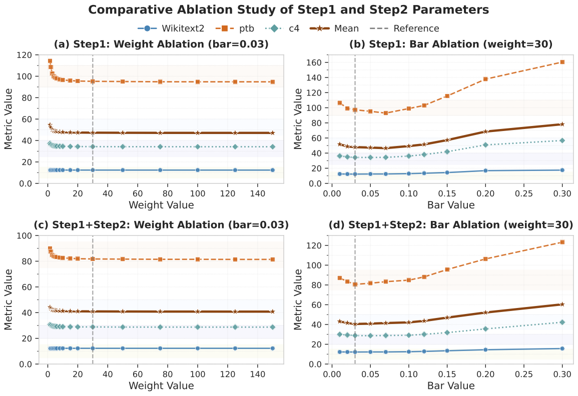

Figure 4: Parameter ablation studies: (a) Step1 weight ablation (bar=0.03), (b) Step1 bar ablation (weight=30), (c) Full DipSVD weight ablation (bar=0.03), (d) Full DipSVD bar ablation (weight=30). Dashed lines mark reference valus (weight=30, bar=0.03).

4.4.2 Modular Sensitivity Study

We conduct ablation studies to evaluate our DipSVD method by: (1) Isolating Step1 (local importance protection with uniform layer compression ratios), (2) Isolating Step2 (global importance protection with uniform SVD compression) and (3) Combining Step1 and Step2 (full DipSVD). This comparison quantifies the impact of local importance protection (Step1) and how global importance protection (Step2) enhances the base performance achieved by local compression (Step1).

From Tab.6, we observe that: (1) Both DipSVD (Step1) and DipSVD (Step2) outperforms SVD-LLM on all datasets, confirming the effectiveness of our proposed local and global importance protection; (2) The full DipSVD (Step1+Step2) further reduces perplexity substantially, demonstrating the synergistic effect of combining both protection strategies in SVD compression.

Method	Hyper	WikiText2	PTB	C4
Vanilla	None	6.7836	30.853	9.2064
SVD-LLM	None	12.4212	124.6766	39.5712
DipSVD (Step1)	weight 30 bar 0.03	12.2663	97.2792	34.2737
DipSVD (Step2)	None	12.1868	94.9336	31.7530
DipSVD (Step1+Step2)	weight 30 bar 0.03	12.1578	80.6209	28.8224

Table 6: Perplexity of compressed Vicuna-7B: DipSVD (Step1) preserves local importance only, DipSVD (Step2) preserves global importance only, while DipSVD (Step1+Step2) preserves both aspects.

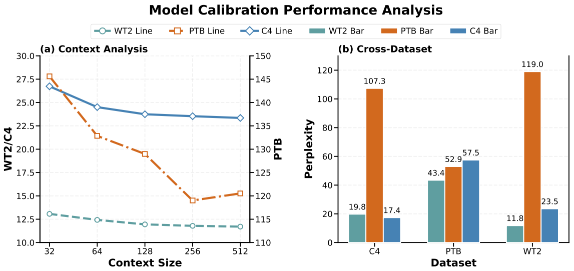

Figure 5: Ablation study of calibration datasets: (a) Perplexity with varying sizes of Wikitext2 calibration data, (b) Perplexity with different calibration datasets.

4.4.3 Impact of Calibration Data.

We examine the impact of calibration data used for channel-weighted data whitening. Fig.5 shows the performance of compressed Vicuna-7B when changing the choice and size of the calibration data. The results show that increased context size will improve the final performance of the compression model to a certain extent, and will work best when the test set is consistent with the calibration set.

5 Conclusion

In this paper, we propose DipSVD, a compression framework that jointly considers both local and global importance factors in large languge models (LLMs) to achieve more efficient and balanced compression. Specifically, we introduce a channel-weighted data whitening technique to preserve the most critical singular vectors within each weight matrix and develop two strategies for determining layer-specific compression ratios, enabling less important layers to absorb a greater portion of the compression burden. Extensive experiments across 10 datasets, 5 models from 3 LLM families, and 4 compression scales consistently demonstrate that DipSVD achieves superior performance compared to existing SVD-based compression methods.

Limitations

While DipSVD demonstrates strong performance across various models, datasets, and compression scales, several limitations remain.

First, the channel-weighted data whitening procedure employs a fixed amplification factor across all layers and channels, which may not optimally adapt to the varing statistical properties of different layers or channel distributions. A more dynamic or data-driven adjustment of the weighting factors could potentially further enhance compression performance.

Second, the assignment of layer-wise compression ratios in DipSVD relies on handcrafted importance metrics based on Fisher sensitivity and effective rank. While this design is computationally efficient and effective in practice, it may not fully capture complex inter-layer dependencies or feature interactions. Developing more sophisticated global importance modeling techniques could potentially lead to further improvements.

Nevertheless, despite these limitations, DipSVD consistently outperforms existing SVD-based compression methods across diverse evaluation settings, demonstrating its robustness and practical effectiveness.

References

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *CoRR*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *Preprint*, arXiv:1905.13319.

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.

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.

Jianfei Chen, Lianmin Zheng, Zhewei Yao, Dequan Wang, Ion Stoica, Michael Mahoney, and Joseph Gonzalez. 2021. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning*, pages 1803–1813. PMLR.

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models. *arXiv preprint arXiv:2305.14323*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

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*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Xuan Ding, Yao Zhu, Yunjian Zhang, and Chuanlong Xie. 2025. A sliding layer merging method for efficient depth-wise pruning in llms. *arXiv preprint arXiv:2502.19159*.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. 2024. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations. *arXiv preprint arXiv:2402.12348*.

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.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and

5 others. 2024. A framework for few-shot language model evaluation.

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

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*.

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. *arXiv preprint arXiv:2402.02834*, 11.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*.

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

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

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.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018a. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018b. Can a suit of armor conduct electricity? a new dataset for open book question answering. *Preprint*, arXiv:1809.02789.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

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

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, pages 1–27.

Xin Wang, Zhongwei Wan, Arvin Hekmati, Mingyu Zong, Samiul Alam, Mi Zhang, and Bhaskar Krishnamachari. 2024a. Iot in the era of generative ai: Vision and challenges. *IEEE Internet Computing*.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024b. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

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

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

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*.

Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. 2024. Investigating layer importance in large language models. *arXiv preprint arXiv:2409.14381*.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, and 1 others. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

A Pseudocode of DipSVD

Algorithm 1 shows the pseudocode of DipSVD. Before compression, DipSVD randomly collects a small amount of sentences as the calibration data C, it then runs the truncation-aware data whitening process as shown in Algorithm 2 to obtain the set of whitening matrix Set_S for the weight to compress. After that, it runs the SVD and truncation with Set_S on each weight matrix in the LLM. Before formally compressing the model, it is necessary to use heuristic methods or Bayesian optimization methods to obtain the compression ratios of different layers, as shown in Algorithm 3.

B Analysis of model computational complexity

Computational complexity mainly depends on the model structure, parameter quantity, sequence length and hardware implementation. This section takes Vicuna-7b model as an example to discuss the computational gain brought by the DipSVD method at a 40% compression ratio.

The DipSVD method achieves significant computational savings in Vicuna-7B through structured low-rank approximation of weight matrices. For a given weight matrix $W \in \mathbb{R}^{m\times n}$, we decompose it via truncated SVD as $W \approx U_l \Sigma_k V_k^T$, where $k \ll \min(m, n)$ is the target rank detemined by Bayesian optimization under a 60% parameter budget constraint ($mk + kn = 0.6mn$). This decomposition transforms the original matrix multiplication WX (computational complexity $C_{WX} = m \cdot n \cdot p$) into a two-step operation: (1) $\hat{X} = \Sigma_k V_k^T X$ (knp FLOPs) followed by (2) $U_k \hat{X}$ (mkp FLOPs), yielding total complexity $C_{UV} = k \cdot p(m + n)$. Substituting the optimal rank $k = \frac{0.6mn}{m+n}$ derived from the parameter constraint, we obtain a theoetical FLOPs reduction ratio of $r = 1 - \frac{C_{UV}}{C_{WX}} = 40\%$.

When applied to Vicuna-7B's 32-layer Transformer architecture, this approach demonstrates three key advantages: (1) The self-attention module's Q/K/V projections (original $3Ld^2$ FLOPs) reduce to $2Ldk$ operations, while the FFN layers' dense matrices ($8Ld^2$ FLOPs) compress to $4Ldk$ operations, where $d = 4096$ and $k \approx 0.6d$. (2) The 40% parameter reduction decreases memory bandwidth pressure, with model size shrinking from 14GB to 8.4GB in FP16 format. Fig.6 presents the throughput and inference speed of Vicuna-7B model compressed by DipSVD across varying batch sizes and sequence lengths. As anticipated, higher compression ratios yield measurable improvements in both throughput and inference speed.

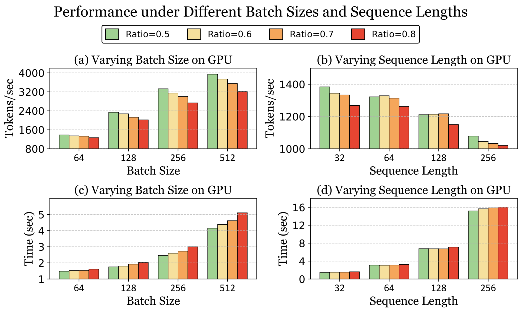

Figure 6: Inference efficiency on Vicuna-7B under different batch sizes and sequence lengths.

C Supplementary Experiment Results

C.1 Detailed performance

Tab.7–11 present a comprehensive comparison between our proposed DipSVD method and existing baselines across five foundational models. The results clearly demonstrate the effectiveness of our approach, with evaluations on three language generation datasets (measured by perplexity) and seven classification tasks (measured by accuracy). The consistent superiority of DipSVD across all benchmarks highlights its robustness and generalization capability.

C.2 Contents Generated from the compressed model

In this section, we present sample outputs generated by our compressed LLaMA-7B model using th proposed DipSVD method at varying compression ratios. As shown in Tab.12, our DipSVD method can maintain high-quality text generation across different compression levels, showcasing its robustness in model compression.

Algorithm 1 Pseudocode of DipSVD

```
Input: Original model $M$, Target compression ratio $R$
Output: Compressed model $M'$
Procedure: DipSVD($M$, $R$)
Randomly collect several sentences as the calibration data $C$
$S \leftarrow$ CHANNEL-WEIGHTED DATA WHITENING$(M, C)$        ▷ Extract the whitening matrix
$\{k_1, k_2, ..., k_L\} \leftarrow$ LAYER-SPECIFIC COMPRESSION$(M)$        ▷ Extract the layer specific compression ratios
for each layer $l \in \{1, ..., L\}$ do
    $W_l \leftarrow$ GetLayerWeights$(\mathcal{M}, l)$        ▷ Obtain the set of weights in $M$ to compress
    $U_l, \Sigma_l, V_l \leftarrow$ SVD$(W_l S_l)$        ▷ Apply singular value decomposition on W
    $k_l \leftarrow \arg\min_k \left( \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum \sigma_i^2} \geq k_l \right)$
    $\Sigma_l' \leftarrow$ diag$(\sigma_1, ..., \sigma_{k_l}, 0, ..., 0)$        ▷ Truncate the smallest singular values in $\Sigma$
    $W_l' \leftarrow U_l \Sigma_l' V_l S_l^{-1}$
    Replace $W_l$ with $W_l'$ in $M'$
end for
return $M'$
End Procedure
```

Algorithm 2 Pseudocode of Channel-weighted data whitening

```
Input: Original model $M$; Calibration Data $C$; Diagonal scaling matrix $D$
Output: Set of whitening matrices in $M$ for the weight to compress Set$_S$
Procedure: CHANNEL-WEIGHTED DATA WHITENING($M$,$C$)
Set$_S \leftarrow \phi$        ▷ Initialize the set of whitening matrices
Set$_W \leftarrow M$        ▷ Obtain the set of weights in $M$ to compress
for $W \in$ Set$_S$ do
    $X \leftarrow M(W, C)$        ▷ Obtain the input activation of the weight matrix $W$
    $\tilde{X}^T \tilde{X} \leftarrow D^T X^T X D$        ▷ Obtain the input activation of the weight matrix $W$
    $S \leftarrow$ SVD$(\tilde{X}^T \tilde{X})$        ▷ Apply singular value decomposition on $\tilde{X}^T \tilde{X}$
    $\Sigma_1 \leftarrow$ Trunc.$(\Sigma)$        ▷ Truncate the smallest singular values in $\Sigma$
    Set$_S \leftarrow S \bigcup$ Set$_S$        ▷ Store the whitening weight matrix in the set
end for
return Set$_S$
End Procedure
```

Algorithm 3 Pseudocode of Layer-specific Compression

Input: Original model M; Input activation x; Target compression ratio R;
Parameters: Global importance preservation method $m \in \{\text{Bayesian}, \text{Heuristic}\}$, Energy threshold $\tau \in (0, 1)$ (default: 0.95), Trade-off parameter $\beta \in [0, 1]$ (default: 0.3)
Output: A list of allocated compression ratios $\{k_1, ..., k_L\}$

```
Randomly collect several sentences as the calibration data $C$
if $m$ = Heuristic then
    $S \leftarrow \text{FisherSensitivity}(M, C)$                                  ▷ Algorithm 4
    $U \leftarrow \text{EffectiveRank}(M, C, \tau)$                                 ▷ Algorithm 5
    Normalize metrics: $\tilde{S} \leftarrow \frac{S-\min(S)}{\max(S)-\min(S)} + \epsilon\ \tilde{U} \leftarrow \frac{U-\min(U)}{\max(U)-\min(U)} + \epsilon$
    Compute combined importance: $W \leftarrow \tilde{S}^{\beta} \circ \tilde{U}^{1-\beta}$
    Allocate ratios: $\{k_1, ..., k_L\} \leftarrow \text{ProportionalAllocation}(W, R)$
else
    Initialize Bayesian Optimizer: $\mathcal{B} \leftarrow \text{BO}(\text{domain} = [0.25, 1]^L, \text{acq} = \text{EI})$
    for $t \leftarrow 1$ to $T$ do
        $\mathbf{r}_t \leftarrow \mathcal{B}.\text{query}()$
        $M' \leftarrow \text{Compress}(M, \mathbf{r}_t)$
        $\text{score} \leftarrow -\text{Perplexity}(M', C)$
        $\mathcal{B}.\text{update}(\mathbf{r}_t, \text{score})$
    end for
    $\{k_1, ..., k_L\} \leftarrow \mathcal{B}.\text{best\_params}()$
end if
return $\{k_1, ..., k_L\}$
```

Algorithm 4 Fisher Sensitivity Computation

```
function COMPUTEFISHERSENSITIVITY($M, C$)
    Initialize $S \leftarrow \mathbf{0}_L$
    for each batch $B \subset C$ do
        Compute gradients $\nabla\mathcal{L}$ via backpropagation
        for each layer $l \in \{1, ..., L\}$ do
            $s_l \leftarrow \sum_{p \in \theta_l} \frac{\|\nabla_p \mathcal{L}\|_2}{\|p\|_2 + \epsilon}$
            $S[l] \leftarrow S[l] + s_l$
        end for
    end for
    Apply segmented normalization to $S$
    return $S^{-1}$                                                    ▷ Inverse sensitivity
end function
```

Algorithm 5 Effective Rank Computation

```
function ComputeEffectiveRank($M, \mathcal{C}, \tau$)
    Initialize $U \leftarrow \mathbf{0}_L$
    for each batch $B \subset C$ do
        Get hidden states $\{h_1, ..., h_L\}$
        for each layer $l \in \{1, ..., L\}$ do
            $H_l \leftarrow \text{reshape}(h_l, [-1, d_l])$
            $\{\sigma_i\} \leftarrow \text{SVD}(H_l)$
            $k \leftarrow \min\{k' | \sum_{i=1}^{k'} \sigma_i^2 \geq \tau \sum \sigma_i^2\}$
            $U[l] \leftarrow U[l] + k$
        end for
    end for
    $U \leftarrow \frac{U}{|\mathcal{C}|/\text{batch\_size}}$
    return z-score($U$)
end function
```

Radio	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
0.8	ASVD	8.759	**12.705**	**10.833**	**0.296**	0.671	**0.694**	**0.505**	**0.340**	**0.750**	0.233	**0.498**
	FWSVD	9.280	14.525	11.880	0.266	**0.679**	0.646	0.472	0.329	0.742	0.228	0.480
	SVD-LLM	**7.894**	16.848	16.118	0.258	0.623	0.652	0.433	0.310	0.687	0.234	0.457
	Ours	7.949	15.597	14.074	0.268	0.633	0.648	0.454	0.328	0.710	**0.238**	0.468
0.7	ASVD	95.268	200.937	86.269	0.186	0.379	0.557	0.333	0.242	0.607	0.218	0.360
	FWSVD	33.001	53.587	38.240	0.186	0.507	0.572	0.343	0.242	0.632	0.217	0.386
	SVD-LLM	9.526	28.967	26.390	0.242	0.509	0.570	0.352	0.269	0.630	0.227	0.400
	Ours	**9.427**	**22.270**	**19.909**	**0.242**	**0.602**	**0.640**	**0.405**	**0.296**	**0.661**	**0.230**	**0.440**
0.6	ASVD	9111.411	19425.612	8676.642	0.158	0.286	0.486	0.267	0.210	0.538	0.204	0.307
	FWSVD	199.142	332.344	255.026	0.158	0.348	0.526	0.275	0.195	0.571	0.211	0.326
	SVD-LLM	13.854	63.864	57.281	0.208	0.455	0.566	0.323	0.240	0.598	0.217	0.372
	Ours	**12.760**	**46.951**	**34.352**	**0.222**	**0.503**	**0.613**	**0.358**	**0.277**	**0.640**	**0.224**	**0.405**
0.5	ASVD	37479.324	57294.849	37767.010	0.124	0.263	0.508	0.256	0.212	0.519	0.202	0.298
	FWSVD	4622.404	8861.445	9240.634	0.124	0.277	0.508	0.260	0.207	0.535	0.210	0.303
	SVD-LLM	26.864	191.380	153.840	0.160	0.361	0.540	0.288	0.207	0.570	0.216	0.334
	Ours	**20.983**	**116.404**	**94.796**	**0.192**	**0.382**	**0.559**	**0.306**	**0.223**	**0.590**	**0.229**	**0.354**

Table 7: Overall Performance of LLama-7B.

Radio	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
0.8	ASVD	13.723	72.029	**18.261**	0.264	0.641	0.624	**0.440**	**0.347**	**0.701**	0.237	**0.465**
	FWSVD	15.312	75.042	19.945	0.238	0.611	**0.627**	0.401	0.309	0.695	0.233	0.445
	SVD-LLM	**9.942**	71.366	23.358	0.252	0.579	0.598	0.401	0.315	0.666	0.229	0.434
	Ours	9.952	**56.869**	19.722	**0.268**	**0.650**	0.598	0.431	0.345	0.689	**0.247**	0.461
0.7	ASVD	91.388	415.615	136.157	0.158	0.335	0.503	0.287	0.208	0.556	0.205	0.322
	FWSVD	43.690	239.318	64.753	0.172	0.459	0.545	0.312	0.224	0.613	0.221	0.364
	SVD-LLM	12.416	124.506	39.528	0.244	0.506	0.570	0.353	0.270	0.629	0.228	0.400
	Ours	**12.144**	**81.089**	**28.837**	**0.248**	**0.573**	**0.597**	**0.384**	**0.293**	**0.659**	**0.232**	**0.427**
0.6	ASVD	1580.427	3069.448	1735.991	0.126	0.273	0.524	0.259	0.224	0.527	0.215	0.307
	FWSVD	347.362	1711.730	461.874	0.128	0.293	0.519	0.267	0.209	0.554	0.212	0.312
	SVD-LLM	18.347	261.100	77.706	0.188	0.430	0.542	0.314	0.238	0.590	0.217	0.360
	Ours	**17.085**	**142.752**	**49.183**	**0.194**	**0.469**	**0.580**	**0.345**	**0.259**	**0.613**	**0.231**	**0.384**
0.5	ASVD	22934.960	28252.915	24201.540	0.146	0.249	0.508	0.258	0.213	0.512	0.185	0.296
	FWSVD	4449.084	9353.528	4421.253	0.128	0.273	0.501	0.262	0.210	0.533	0.201	0.301
	SVD-LLM	35.569	615.591	185.780	0.158	0.338	0.525	0.286	0.231	0.567	0.225	0.333
	Ours	**27.807**	**375.093**	**111.996**	**0.170**	**0.365**	**0.543**	**0.300**	**0.220**	**0.571**	**0.221**	**0.342**

Table 8: Overall Performance of Vicuna-7B.

Radio	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
0.8	ASVD	12.727	17.215	**16.851**	0.288	0.654	0.650	**0.486**	0.325	**0.742**	0.237	0.483
	FWSVD	15.578	23.550	24.258	0.238	0.617	0.643	0.411	0.274	0.714	0.242	0.448
	SVD-LLM	9.128	19.106	21.004	0.286	0.636	0.650	0.431	0.330	0.711	0.243	0.470
	Ours	**8.805**	**15.890**	17.732	**0.290**	**0.665**	**0.663**	0.450	**0.342**	0.721	**0.256**	**0.484**
0.7	ASVD	85.169	87.709	79.853	0.154	0.390	0.516	0.312	0.213	0.610	0.210	0.344
	FWSVD	68.416	99.775	118.319	0.142	0.406	0.551	0.296	0.194	0.595	0.220	0.344
	SVD-LLM	10.841	30.747	32.622	0.260	0.589	0.609	0.384	0.283	0.670	0.232	0.432
	Ours	**9.895**	**20.977**	**22.558**	**0.276**	**0.628**	**0.631**	**0.415**	**0.312**	**0.700**	**0.239**	**0.457**
0.6	ASVD	3806.825	7580.528	4355.394	0.140	0.298	0.494	0.267	0.202	0.546	0.209	0.308
	FWSVD	202.822	265.391	325.196	0.126	0.309	0.499	0.267	0.184	0.545	0.215	0.306
	SVD-LLM	14.449	55.803	58.199	0.228	0.529	0.578	0.336	0.241	0.626	0.225	0.395
	Ours	**12.077**	**32.890**	**35.540**	**0.250**	**0.572**	**0.624**	**0.366**	**0.285**	**0.653**	**0.233**	**0.426**
0.5	ASVD	64971.820	99927.992	57731.498	0.128	0.262	0.498	0.263	0.213	0.515	0.201	0.297
	FWSVD	600.397	890.699	774.628	0.132	0.281	0.478	0.264	0.195	0.531	0.205	0.298
	SVD-LLM	22.660	132.187	117.580	0.192	0.427	0.538	0.300	0.220	0.591	0.213	0.355
	Ours	**17.960**	**79.482**	**73.707**	**0.194**	**0.468**	**0.554**	**0.316**	**0.222**	**0.609**	**0.223**	**0.369**

Table 9: Overall Performance of Deepseek-7B.

Radio	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
0.8	ASVD	6.743	**10.407**	**9.123**	**0.330**	0.732	**0.707**	0.540	0.431	0.771	**0.251**	**0.538**
	FWSVD	6.938	11.231	9.499	0.312	**0.737**	0.696	0.518	0.393	0.769	0.258	0.526
	SVD-LLM	**6.575**	12.194	12.811	0.302	0.683	0.684	0.470	0.356	0.725	0.265	0.498
	Ours	6.649	11.383	11.418	0.306	0.681	0.692	0.490	0.369	0.734	**0.258**	0.503
0.7	ASVD	17.648	32.963	20.866	0.218	0.551	0.611	0.398	0.288	0.690	0.217	0.425
	FWSVD	12.963	22.123	18.509	0.248	0.632	0.640	0.403	0.294	**0.707**	0.229	0.450
	SVD-LLM	**7.618**	17.823	18.825	0.276	0.619	0.672	0.415	0.300	0.671	0.242	0.457
	Ours	7.697	**15.681**	**14.614**	**0.284**	**0.641**	**0.656**	**0.449**	**0.328**	0.694	**0.249**	**0.472**
0.6	ASVD	201.027	286.850	183.898	0.148	0.336	0.518	0.293	0.197	0.579	0.215	0.327
	FWSVD	45.150	75.662	64.610	0.166	0.431	0.552	0.305	0.219	0.607	0.226	0.358
	SVD-LLM	9.836	34.222	33.328	0.222	0.521	0.639	0.355	0.248	0.637	0.228	0.407
	Ours	**9.575**	**25.782**	**21.716**	**0.230**	**0.548**	**0.644**	**0.402**	**0.283**	**0.661**	**0.233**	**0.429**
0.5	ASVD	11445.274	13304.711	10897.571	0.116	0.270	0.490	0.267	0.221	0.535	0.206	0.301
	FWSVD	193.531	275.487	245.409	0.126	0.304	0.518	0.270	0.176	0.550	0.213	0.308
	SVD-LLM	14.984	89.288	68.417	**0.194**	0.420	0.582	0.314	0.224	0.588	0.220	0.363
	Ours	**13.526**	**74.449**	**45.166**	0.192	**0.447**	**0.603**	**0.333**	**0.229**	**0.612**	**0.229**	**0.378**

Table 10: Overall Performance of LLama-13B.

Radio	Method	WikiText-2	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average
0.8	ASVD	10.261	135.308	14.523	0.276	0.684	0.663	0.466	0.368	**0.739**	0.259	0.493
	FWSVD	12.636	152.403	19.540	0.230	0.657	0.624	0.412	0.312	0.717	0.237	0.456
	SVD-LLM	8.035	82.811	18.269	**0.314**	**0.687**	0.671	0.456	**0.378**	0.727	**0.264**	**0.500**
	Ours	**7.892**	**36.405**	**14.225**	0.306	0.666	**0.674**	**0.471**	0.357	0.718	0.262	0.493
0.7	ASVD	28.309	637.196	39.799	0.220	0.528	0.574	0.340	0.250	0.669	0.226	0.401
	FWSVD	32.715	310.304	47.408	0.188	0.506	0.564	0.317	0.234	0.647	0.223	0.383
	SVD-LLM	9.616	145.715	29.204	**0.284**	**0.620**	0.641	0.390	0.300	0.656	**0.250**	0.449
	Ours	**9.070**	**49.948**	**19.203**	0.268	0.610	**0.657**	**0.419**	**0.320**	**0.688**	0.243	**0.458**
0.6	ASVD	189.392	1601.441	190.550	0.128	0.310	0.518	0.276	0.202	0.572	0.214	0.317
	FWSVD	106.356	668.389	146.253	0.130	0.307	0.510	0.278	0.190	0.550	0.218	0.312
	SVD-LLM	13.221	324.892	58.335	**0.244**	0.506	0.600	0.333	0.237	0.598	**0.232**	0.393
	Ours	**12.138**	**73.798**	**32.224**	0.230	**0.511**	**0.609**	**0.358**	**0.273**	**0.640**	0.227	**0.407**
0.5	ASVD	1220.318	3922.871	1080.129	0.144	0.275	0.485	0.262	**0.228**	0.536	0.209	0.306
	FWSVD	304.176	1475.578	395.776	0.124	0.284	0.489	0.264	0.207	0.540	0.202	0.301
	SVD-LLM	23.607	699.598	140.584	0.172	0.372	0.563	0.296	0.218	0.563	**0.220**	0.343
	Ours	**18.150**	**148.365**	**71.652**	**0.184**	**0.404**	**0.563**	**0.311**	0.215	**0.582**	0.217	**0.354**

Table 11: Overall Performance of Vicuna-13B.

Score	Description
What is the responsibility of the AI assistant?	
1	AI assistants are responsible for understanding and responding to the user's requests. They also need to be able to make decisions and take action on behalf of the user. For example, if a user asks Alexa to turn on the lights in the living room, Alexa needs to understand what "turn on" means and then activate the light bulbs accordingly.
0.8	The AI assists you in your work. For example, if you are driving a car, it helps you to find parking spaces. If you're looking for a book, it shows you where the book is located.
0.7	The AI assists the player in various ways. For example, if the player has not completed a particular task, such as collecting an item or completing a puzzle, they are prompted to do so by their AI.
0.6	How AI assistants can be used by AI researchers and AI scientists. The AI Assistant in AI has made AI more important as AI evolves into AI for AI.
0.5	The AI assistant, known as a robot, operates autonomously in an autonomic environment. Robots can operate autonomically and have their own autonomic system.
What is the universe? Explain this concept.	
1	The Universe is everything that exists. It includes all of the planets, stars, galaxies, and other celestial bodies in our solar system, as well as invisible elements like dark matter and dark energy.
0.8	The universe is everything that is, will be, or may be. It includes all matter, energy, antimatter, stars, galaxies, planetary systems, interstellar gases, asteroids, comets, and nebulae.
0.7	The universe is made up of billions of galaxies, which contain hundreds of millions of stars. Each star has its own orbit around the center of the galaxy.
0.6	The universe is a collection of galaxies bound by interstellar gases. A single galaxy may contain up to 100 million stars, and hundreds of such galaxies exist.
0.5	The universe may not be defined solely by cosmological principles like astronomy and cosmology.

Table 12: Sample outputs generated by DipSVD of LLaMA-7B at varing compression ratios.