MTL-SVD: Minimizing Singular Value Truncation Loss for Large Language Model Compression

Anonymous ACL submission

1 Abstract

Despite significant advancements, the practical deployment of Large Language Models (LLMs) is often hampered by their immense sizes, highlighting the need for effective compression techniques. Singular Value Decomposition (SVD) emerges as a promising method for compressing LLMs. However, existing SVD-based compression approaches suffer from substantial truncation losses, leading to severe performance degradation in the compressed models. In this work, we introduce MTL-SVD, a novel SVD-based LLM compression method that **M**inimizes the **T**runcation **L**oss in **SVD** compression with two key strategies. First, MTL-SVD employs dynamic compression ratio allocation to effectively balance the extremely large truncation loss across different layers. Second, it implements loss-minimized weight truncation to ensure that the truncated singular values result in the lowest possible truncation loss. We evaluate MTL-SVD on 10 datasets and five models at various scales. Our results demonstrate that MTL-SVD outperforms current state-of-the-art methods.

2 Introduction

Despite the outstanding performance that large Language Models (LLMs) have exhibited in various tasks (Zhao et al., 2023; Gozalo-Brizuela and Garrido-Merchán, 2023), the significant resource requirements severely limit their widespread accessibility (Wan et al., 2023; Wang et al., 2024a; Zhou et al., 2024). Model compression (Zhu et al., 2023) is one of the most effective approaches to mitigate these resource demands. To avoid the necessity for resource-heavy retraining, LLM compression is often conducted in a post-training manner. Techniques such as LLM quantization (Yuan et al., 2024; Huang et al., 2024), unstructured pruning (Frantar and Alistarh, 2023), and structured pruning(Ma et al., 2023; Ashkboos et al., 2024; Zhong et al., 2024) have been intensively studied.

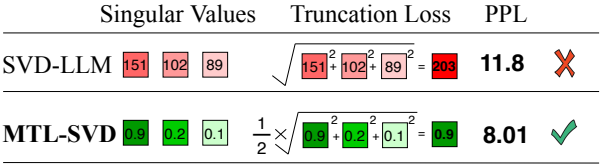


Figure 1: Comparison of MTL-SVD with state-of-the-art SVD-based LLM compression method SVD-LLM. We randomly select a weight matrix from LLaMA-3 8B and compare its singular values and the truncation loss. Other weight matrices also share the similiar trend. The PPL is measured under 20% compression ratio of LLaMA-3 8B by different compression methods.

Low-rank approximation, such as Singular Value Decomposition (SVD) is also a commonly used technique for compressing LLMs. Compared with the other types of post-training LLM compression techniques, such as quantization and unstructured pruning, SVD compression is much more hardware-friendly. Many SVD-based LLM compression methods such as FWSVD (Hsu et al., 2022), ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2024b) have recently been proposed. However, these methods often exhibit severe performance degradation. The key reason is due to the inability of these methods to ensure low truncation loss, even under a low compression ratio[1]. As a result, given the input activation, the output between the compressed weight matrices and the original ones in the LLM will be significantly different, making the compressed LLM behave much poorer than the original LLM. For a better demonstration, we analyzed the truncation loss and the final performance of the state-of-the-art SVD-based LLM compression methods SVD-LLM (Wang et al., 2024b) on LLaMA-3 8B. As shown in Figure 1, even under 20% compression ratio, SVD-LLM still results in a

[1]The compression ratio in our paper refers to the percentage of parameter reduction achieved through compression.

huge truncation loss and suffer from a severe drop in perplexity.

In this paper, we propose a SVD-based post-training LLM compression method, which we refer to as MTL-SVD, which effectively addresses the two fundamental issues of existing methods. Specifically, to address the issue of extremely large truncation losses in certain weight matrices, MTL-SVD dynamically allocates compression ratios among the weight matrices across different layers for balance. MTL-SVD. To further reduce the truncation loss of the weight matrices, MTL-SVD injects an optimizing matrix during the SVD truncation, which simplifies the calculation of the upper-bound of truncation loss. Through minimizing this upper-bound, the truncation loss could also be reduced significantly.

We evaluate the generability of MTL-SVD in 10 datasets on 5 different LLMs with various scales. Our results demonstrate the superiority of MTL-SVD with three key findings as follows:

- MTL-SVD consistently outperforms state-of-the-art SVD-based LLM compression methods on 5 different LLMs and across all 10 datasets covering various language modeling, classification and generation tasks.
- MTL-SVD outperforms state-of-the-art structured pruning-based LLM compression methods with up to 28% lower perplexity under 7GB memory budget. When compressing a 16-bit LLM, MTL-SVD achieves performance comparable to using 1-bit post-training quantization techniques under the same memory budget. Moreover, when compressing a 2-bit LLM, MTL-SVD even surpasses the performance of the state-of-the-art 1-bit quantization with up to 69% lower perplexity.
- LLMs compressed by MTL-SVD are able to achieve inference speedup on real hardware. In particular, compared to the original LLMs, LLMs compressed by MTL-SVD are able to achieve at most 2.71$\times$ speedup on the single NVIDIA A100 GPU.

3 Related Work

3.1 Large Language Model Compression

Large Language Models (LLMs) typically contain billions of parameters, making traditional model compression techniques impractical due to the need for resource-intensive retraining. To address this, post-training methods that bypass retraining during compression have been developed. These methods generally fall into four categories: unstructured pruning, structured pruning, quantization, and low-rank approximation. Unstructured pruning (Frantar and Alistarh, 2023) sets individual weight value to zero without changing the overall architecture. However, its irregular sparsity is only feasible for speedups or memory savings on certain hardware. In contrast, structured pruning (Ma et al., 2023; Ashkboos et al., 2024; Zhong et al., 2024) removes entire channels from LLMs, simplifying hardware implementation, but often suffering from accuracy degradation. Quantization reduces the precision of the weight matrices for compression. However, it often fails to provide the desired inference speedups (Lin et al., 2024) and offers a limited range of compression options—typically between 2 to 8 bits—which hinders optimal memory utilization. Recent efforts (Yuan et al., 2024; Huang et al., 2024) have explored 1-bit post-training quantization. Nevertheless, these approaches still suffer from accuracy drop, indicating that 1-bit quantization is still challenging in LLM compression.

3.2 SVD for LLM Compression

Singular Value Decomposition (SVD) reduces matrix sizes by truncating the smallest singular values. It then constructs two smaller, lower-rank matrices as the estimation with the remaining singular values (Golub et al., 1987). SVD is also feasible for LLM compression (Hsu et al., 2022) and many SVD-based compression techniques have been proposed. For example, Yuan et al. (2023) proposed ASVD, which scales the weight matrix using a diagonal matrix to normalize the impact of input channels on the weights. Wang et al. (2024b) made further advancement by whitening the input matrix to mitigate its impact on SVD truncation and achieves the state-of-the-art performance. However, all of the existing methods still suffer from the accuracy degradation with huge truncation loss, especially under the high compression ratio.

4 MTL-SVD

Figure 2 provides an overview of MTL-SVD. Specifically, it first groups the weight matrices by type and computes their truncation loss. After that, it dynamically allocates the compression ratio for the weight matrices in each group based on the computed loss. After obtaining all the compression

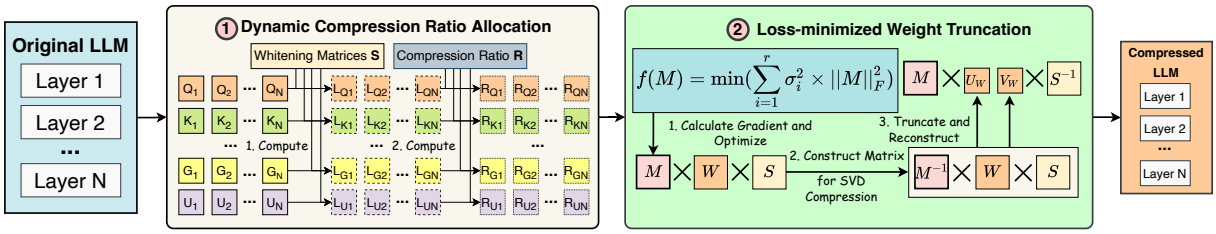


Figure 2: Overview of MTL-SVD.

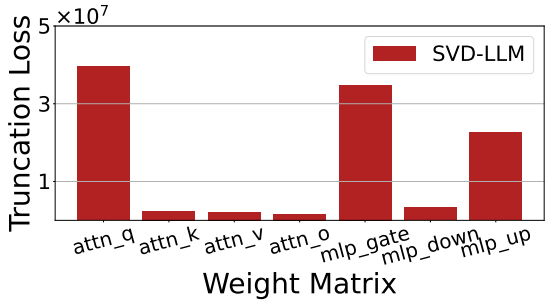


Figure 3: Truncation loss by SVD-LLM of weight matrices in one randomly selected decoding layer in LLaMA-3 8B on WikiText-2 under 50% compression ratio.

ratios for the singular value truncation, MTL-SVD multiplies the weight matrix with an optimizing matrix M to obtain an upper-bound of the truncation loss. In this way, the minimization of loss could be transformed into the minimization of the upper-bound (denoted by $f(M)$ in the figure) by computing the gradient and optimizing the matrix M. Finally, MTL-SVD applies SVD compression and truncation to the matrix updated by M and reconstructs the weights for storage. In the following, we first present our motivation towards the design of MTL-SVD and then describe both dynamic compression ratio allocation and loss-minimized weight truncation in detail.

4.1 Motivation

To ensure better compression performance, many recent post-training SVD-based LLM compression works (Hsu et al., 2022; Yuan et al., 2023; Wang et al., 2024b) attempt to lower the truncation loss L in the form of Frobenius norm as follows during their LLM compression:

$$L = ||WX - W'X||_F \tag{1}$$

where W is the weight matrix of the original LLM, X is the activation of W, and W' is the compressed low-ranking weight matrix.

Although the state-of-the-art method SVD-LLM (Wang et al., 2024b) has theoretically guaranteed the alignment between truncated singular values and the corresponding truncation loss, which reduces the truncation loss and significantly enhances the compressed LLMs' performance, it still fails to ensure the minimization of the truncation loss during the SVD compression. For a better demonstration, we randomly select one of the decoding layers in LLaMA-3 8B and compute the corresponding truncation loss of all weight matrices under 50% compression ratio. As shown in Figure 3, all the weight matrices suffer severely from the huge truncation loss. For the weight matrices in other decoding layers, we also observe a similar trend. As a result, the output $W'X$ computed by the compressed weight matrix W significantly deviates from WX computed by the original weight matrix W, causing the final performance of compressed LLM to drop significantly. Therefore, it is still necessary to design a new method with a better ability to minimize truncation loss.

4.2 Dynamic Compression Ratio Allocation

Motivation: Since different weight matrices in LLMs often exhibit varying levels of redundancy, one of the most effective strategies to reduce truncation loss is to dynamically allocate the compression ratio to balance the extremely large truncation loss. To demonstrate, we measured the truncation loss by the state-of-the-art method SVD-LLM of query weight matrix across different layers in LLaMA-3 8B on WikiText-2 dataset under 50% compression ratio. As shown in Figure 4, the truncation loss varies between layers. For example, the query matrix in the 27th layer has a much higher truncation loss than the 1st layer, indicating that the 27th layer should receive a smaller compression ratio, while the 1th layer can tolerate a larger one. However, existing SVD-based LLM compression methods overlook this dynamic strategy or require resource-intensive steps to determine compression ratios, making them impractical for larger-scale LLMs. Therefore, it is essential to develop a more effective

approach for dynamically allocating compression ratios across different weight matrices to reduce truncation loss in LLMs.

Key Design: The pseudocode of the dynamic compression ratio allocation is provided in Algorithm 1. Specifically, to calculate the truncation loss more efficiently, MTL-SVD applies a whitening matrix S, as done in SVD-LLM (Wang et al., 2024b) to align the truncation loss with singular values. We follow the same procedure in SVD-LLM to construct a calibration set X and compute the whitening matrix S with the Choloesky decompostion of XX^T. In this way, the truncation loss L to truncate the multiple singular values $(\sigma_0, \sigma_1, \cdot\sigma_k)$ could easily be obtained by $\sqrt{\sum_{i=0}^{k} \sigma_i^2}$. Since different types of weight matrices, such as Attention matrices and MLP matrices, perform distinct roles in the LLM, it is inappropriate to consider them together for compression ratio allocation. Therefore, given a target compression ratio R, MTL-SVD first groups the weight matrices into different layers according to their type. The compression ratio is then linearly allocated based on the computed and normalized truncation loss under the target compression ratio R within each group. As shown in Figure 4, after applying the dynamic compression ratio allocation, the extremely large truncation loss meets a significant drops (the blue area) with only a small increase of several small truncation loss (the yellow area).

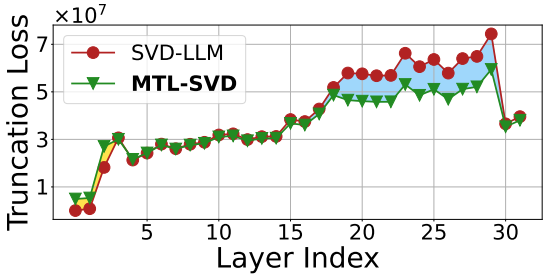


Figure 4: Truncation loss by SVD-LLM and MTL-SVD of the query weight across different layers in LlaMA-3 8B on WikiText-2 under 50% compression ratio.

4.3 Loss-minimized Weight Truncation

Motivation: After obtaining the dynamic compression ratio for every weight matrix in the LLM, our next job is to truncate the weight to its allocated compression ratio. However, even the state-of-the-art method SVD-LLM, still suffer from the huge truncation loss under a small compression ratio. For a better demonstration, we randomly select a weight matrix in LLaMA-3 8B and compute its truncation loss by SVD-LLM on WikiText-2 dataset under different compression ratios. As shown in Figure 5, even when the compression ratio is as low as 20%, SVD-LLM still suffers a huge truncation loss. Therefore, it is necessary to minimize the truncation loss for SVD compression. One of the most intuitive ways is to update the compressed matrices W' using input activation X. However, given that the input activations of the LLM contain tremendous parameters, it is not practical to cache them for usage. Therefore, it is still challenging to design a feasible way for minimizing the truncation loss.

Algorithm 1 Pseudocode of Dynamic Compression Ratio Allocation in MTL-SVD

Input: M: Original LLM
S: A set of whitening matrices
R: Target compression ratio
Output: R_d: A list of allocated compression ratios

procedure DYNAMIC COMPRESSION RATIO ALLOCATION(M, S, R)
  $G \leftarrow \text{Group}(M)$ ▷ Group the weight by types
  $R_d \leftarrow \emptyset$ ▷ Initialize the compression ratio list
  for g **in** G **do**
    $L \leftarrow \emptyset$ ▷ Initialize the loss list in the group
    for w **in** g **do**
      $l \leftarrow \text{Compute_Loss}(w, S(w), R)$
      $L \leftarrow L \cup l$ ▷ Append l to the list L
    end for
    $L \leftarrow 1/\text{Log}(L)$ ▷ Normalize L to log form
    $r \leftarrow L/\text{Sum}(L)$ ▷ Computes r from L
  end for
  $R_d \leftarrow R_d \cup r$ ▷ Append r to the list R_d
  return M''
end procedure

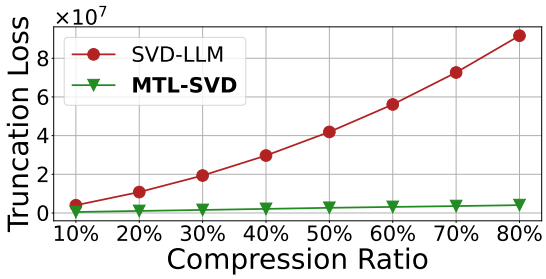


Figure 5: Truncation loss by SVD-LLM and MTL-SVD of the weight matrix randomly selected in LLaMA-3 8B on WikiText-2 under different compression ratios.

Key Design: Given a weight matrix W, the key idea of loss-minimized weight truncation is to multiplies an invertible matrix M and its inverse on the left side of W and the whitening matrix S, which were obtained during the dynamic compression ratio allocation step, on the right side. Instead of decomposing the original matrix W as in vanilla SVD, or decomposing WS as done in

ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2024b), MTL-SVD applies SVD decomposition and singular value truncation on the multiplication result of $M^{-1} \times W \times S$. In this way, we could leverage matrix M to further minimize truncation loss effectively and efficiently without the need to store the huge activation matrix X. Below we provide a theoretical proof to demonstrate why this truncation strategy offers such benefit.

Theorem 4.1. *If M is the invertible matrix and S is the Cholesky decomposition of XX^T, the truncation loss L of truncating smallest singular values $\sigma_1, ..., \sigma_r$ is no more than $\sqrt{||M||_F^2 \sum_{i=1}^{r} \sigma_i^2}$.*

Proof. Since M is the invertible matrix and S is the Cholesky decomposition of XX^T, we have $MM^{-1} = SS^{-1} = I, SS^T = XX^T$, suppose that we compress W to W', the square of truncation loss L is computed as:

$$\begin{aligned} L^2 &= ||WX - W'X||_F^2 \\ &= ||MM^{-1}WSS^{-1}X - MM^{-1}W'SS^{-1}X||_F^2 \\ &= ||MM^{-1}WSS^{-1}X - M\mathbf{SVD}(M^{-1}WS)S^{-1}X||_F^2 \\ &= ||M(M^{-1}WS - \mathbf{SVD}(M^{-1}WS))S^{-1}X||_F^2 \\ &= ||M\sigma_i u_i v_i^T S^{-1}X||_F^2 \end{aligned} \tag{2}$$

Since $||A||_F^2 = \mathbf{trace}(A^T A)$, we have:

$$\begin{aligned} L^2 &= ||WX - W'X||_F^2 = ||M\sigma_i u_i v_i^T S^{-1}X||_F^2 \\ &= \sum_{j=1}^{r}\sum_{i=1}^{r} \sigma_i\sigma_j \mathbf{trace}(Mu_i v_i^T S^{-1}XX^T(S^{-1})^T v_j u_j^T M^T) \\ &= \sum_{j=1}^{r}\sum_{i=1}^{r} \sigma_i\sigma_j \mathbf{trace}(Mu_i v_i^T (S^{-1}XX^T(S^{-1})^T) v_j u_j^T) \\ &= \sum_{j=1}^{r}\sum_{i=1}^{r} \sigma_i\sigma_j \mathbf{trace}(Mu_i v_i^T (S^{-1}SS^T(S^{-1})^T) v_j u_j^T) \\ &= \sum_{j=1}^{r}\sum_{i=1}^{r} \sigma_i\sigma_j \mathbf{trace}(Mu_i v_i^T v_j u_j^T M^T) \end{aligned} \tag{3}$$

Since $v_i^T v_i = u_i^T u_i = 1; v_i^T v_j = u_i^T u_j = 0, ||v_i|| = ||u_i|| = 1, \forall i \neq j$, and $||AB||_F^2 \leq ||A||_F^2 \cdot ||B||_F^2$. Therefore, we have:

$$\begin{aligned} L^2 &= ||WX - W'X||_F^2 \\ &= \sum_{j=1}^{r}\sum_{i=1}^{r} \sigma_i\sigma_j \mathbf{trace}(Mu_i v_i^T v_j u_j^T M^T) \\ &= \sum_{i=1}^{r} \sigma_i^2 \mathbf{trace}(Mu_i u_i^T M^T) \\ &= \sum_{i=1}^{r} \sigma_i^2 ||Mu_i||_F^2 \leq ||M||_F^2 \sum_{i=1}^{r} \sigma_i^2 \end{aligned} \tag{4}$$

Therefore, we have $L \leq \sqrt{||M||_F^2 \sum_{i=1}^{r} \sigma_i^2}$. □

Therefore, we could indirectly minimize the truncation loss through lowering this upper-bound. The gradient of L can be calculated as:

$$\begin{aligned} \nabla_M L^2 &= \nabla_M(||M||_F^2 \cdot \sum_{i=1}^{r} \sigma_i^2) \\ &= \nabla_M ||M||_F^2 \cdot \sum_{i=1}^{r} \sigma_i^2 + ||M||_F^2 \nabla_M \sum_{i=1}^{r} \sigma_i^2 \\ &= \sum_{i=1}^{r} \sigma_i^2 \cdot 2M - 2||M||_F^2 \cdot \sum_{i=1}^{r} \sigma_i (M^{-1})^T U_i V_i^T (WS)^T \end{aligned} \tag{5}$$

where U_i and V_i are the left and right singular vectors corresponding to the i-th singular value σ_i. Therefore, the gradient is easy to obtain, which makes it convenient for us to apply some existing optimization algorithms[2] with the computed gradient to further minimize the truncation loss. Different from directly optimizing W' based on the original truncation loss $L = ||WX - W'X||_F$ that requires storing every large-scale activation matrix X, which is impractical in the post-training LLM compression scenario, optimizing with gradient computed from Equation (5) does not need to store activation, which is more efficient and feasible.

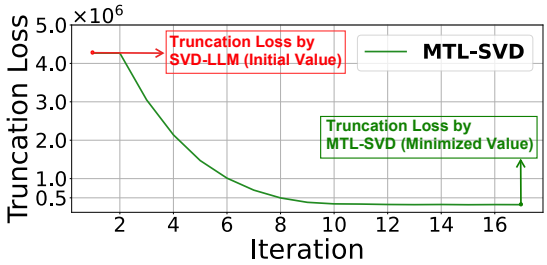


Figure 6: Truncation loss (upper-bound) of the weight matrix in LLaMA-3 8B during the optimization process.

To further improve the optimization efficiency, we initialize M to identity matrix, i.e., $M = I$. Therefore, the initial truncation loss is $\sqrt{\sum_{i=1}^{r} \sigma_i^2}$, which is exactly equal to the truncation loss obtained by SVD-LLM. For a better demonstration, we randomly select one weight matrix in LLaMA-3 8B and plot its upper-bound truncation loss during the optimization process. As shown in Figure 6, the truncation loss drops significantly via optimization, demonstrating the effectiveness of our proposed loss-minimized weight truncation strategy.

5 Experiments and Analysis

Baselines. We compare MTL-SVD against two groups of methods. (1) Three state-of-the-art SVD-

[2] In our paper, we use L-BFGS (Nash and Nocedal, 1991) algorithms as the default

Table 1: Performance of OPT-6.7B, LLaMA-7B, and LLaMA-3 8B compressed by MTL-SVD and baselines under 20% compression ratio on two language modeling datasets (measured by perplexity (↓)), eight common sense reasoning datasets (six measured by both individual and average accuracy (↑), TruthfulQA measured by BLEU score (↑), and GSM8K measured by Exact Match Accuracy (↑)). The best performance is marked in bold. The relative performance gain compared to the best-performing baseline is marked in green inside bracket.

	METHOD	WikiText-2↓	C4↓	Openb.	ARC_e	WinoG.	HellaS.	PIQA	MathQA	**Average↑**	TruthfulQA↑	GSM8K↑
LLaMA-7B	Original	5.68	7.34	0.34	0.75	0.70	0.57	0.79	0.27	0.57	0.30	0.09
	FWSVD	1727	1511	0.09	0.11	0.05	0.08	0.10	0.05	0.08	0.00	0.00
	ASVD	11.14	15.93	0.29	0.53	0.64	0.41	0.68	0.17	0.45	0.21	0.04
	SVD-LLM	7.94	15.84	0.31	0.71	0.68	0.49	0.71	0.22	0.52	0.24	0.06
	MTL-SVD	**7.12** (↓10%)	**10.47** (↓34%)	**0.32**	**0.72**	**0.70**	**0.52**	**0.75**	**0.24**	**0.54** (↑4%)	**0.27** (+0.03)	**0.07** (+0.01)
OPT-6.7B	Original	10.87	12.50	0.28	0.66	0.65	0.50	0.76	0.25	0.52	0.29	0.01
	FWSVD	14559	17898	0.03	0.08	0.02	0.01	0.05	0.01	0.03	0.01	0.00
	ASVD	82	102	0.16	0.41	0.30	0.36	0.61	0.07	0.32	0.09	0.00
	SVD-LLM	16.04	21.27	0.21	0.56	0.59	0.47	0.73	0.21	0.46	0.22	0.00
	MTL-SVD	**13.46** (↓16%)	**17.72** (↓17%)	**0.25**	**0.61**	**0.62**	**0.49**	**0.74**	**0.22**	**0.49** (↑7%)	**0.24** (+0.02)	**0.01** (+0.01)
LLaMA-3 8B	Original	6.14	9.47	0.35	0.80	0.73	0.60	0.80	0.40	0.61	0.49	0.45
	FWSVD	4782	8195	0.01	0.04	0.01	0.02	0.02	0.01	0.02	0.00	0.00
	ASVD	17.55	28.41	0.20	0.59	0.61	0.41	0.69	0.30	0.47	0.37	0.28
	SVD-LLM	11.82	20.05	0.29	0.77	0.64	0.51	0.72	0.30	0.54	0.45	0.31
	MTL-SVD	**8.01** (↓32%)	**11.72** (↓42%)	**0.33**	**0.79**	**0.70**	**0.58**	**0.77**	**0.36**	**0.59** (↑9%)	**0.46** (+0.01)	**0.40**(+0.09)

based LLM compression methods: FWSVD (Hsu et al., 2022),ASVD (Yuan et al., 2023) ,and MTL-SVD (Wang et al., 2024b) (Section 5.1). (2) Other types of LLM compression methods. These include three state-of-the-art pruning-based LLM compression methods: LLM-Pruner (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), and Block-Pruner (Zhong et al., 2024), and two state-of-the-art quantization-based LLM compression methods: PB-LLM (Yuan et al., 2024), and BiLLM (Huang et al., 2024).(Section 5.4).

Models and Datasets. To demonstrate the generability of our method, we evaluate the performance of MTL-SVD on five models at various scales (LLaMA-7B, 13B, 30B, LLaMA3-8B (Touvron et al., 2023), OPT-6.7B (Zhang et al., 2022)) and 10 datasets including two language modeling datasets (WikiText-2 (Merity et al., 2017), and C4 (Raffel et al., 2020)), six classification datasets (OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), Arc_e (Clark et al., 2018), PIQA (Bisk et al., 2020), MathQA (Amini et al., 2019)), and two generation datasets (TruthfulQA (Lin et al., 2022), and GSM8K (Cobbe et al., 2021) with the LM-Evaluation-Harness framework (Gao et al., 2023).

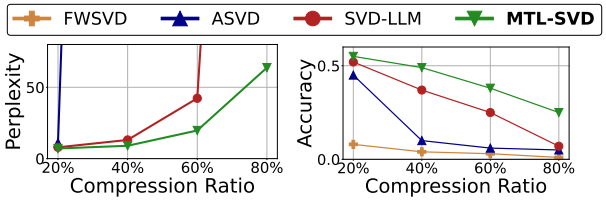
Implementation Details. To ensure a fair comparison, we followed SVD-LLM (Wang et al., 2024b) to randomly select 256 samples from WikiText-2 as the calibration data. The optimization details in loss-minimized weight truncation procedure is provided in Appendix A.2. All of the experiments are conducted on A100 GPUs.

5.1 Performance Comparison

To examine the generability of MTL-SVD, we compare it with state-of-the-art SVD-based LLM compression methods from four aspects: (1) performance on different LLMs, (2) performance on LLMs with larger scales, (3) performance under different compression ratios, and (4) compression speed analysis (provided in Appendix A.1).

Performance on Different LLMs. We compare the performance between MTL-SVD and the baselines on three different LLMs, including LLaMA-7B, OPT-6.7B, and LLaMA-3 8 under 20% compression ratio on two language modeling and six classification, and two generation datasets. As shown in Table 1, MTL-SVD consistently achieves better and more stable performance than other baselines on all three LLMs and all datasets. In particular, it achieves at most 42% perplexity reduction and 9% accuracy improvement with better generation ability compared to the prior state-of-the-art method SVD-LLM on LLaMA-3 8B.

Performance on LLMs with Larger Scales. We next compare the performance between MTL-SVD and the baselines on LLaMA-13B, and LLaMA-30B under 20% compression ratio on WikiText-2 and six classification datasets. As shown in Table 2, MTL-SVD consistently outperforms all baselines



(a) Perplexity (b) Average Accuracy

Figure 7: Perplexity on WikiText-2 and average accuracy on six classification datasets of LLaMA-7B compressed by MTL-SVD and other SVD-based LLM compression baselines under 20% to 80% compression ratios. The perplexity of FWSVD and ASVD are larger than 100, thus get evicted from the figure.

on both of the two model sizes.

Table 2: Perplexity (↓) on WikiText-2 and average accuracy (↑) of six classification datasets of LLaMA-13B and LLaMA-30B under 20% compression ratio.

	LLaMA-13B		LLaMA-30B	
METHOD	Perplexity↓	Accuracy↑	Perplexity↓	Accuracy↑
Original	5.09	0.59	4.10	0.61
FWSVD	15.98	0.43	20.54	0.42
ASVD	6.74	0.54	22.71	0.44
SVD-LLM	6.61	0.55	5.63	0.57
MTL-SVD	**5.46** (↓17%)	**0.56** (↑2%)	**4.71** (↓16%)	**0.60** (↑5%)

Performance under Different Compression Ratios. We finally compare the performance between MTL-SVD and the baselines on LLaMA-7B under compression ratio ranging from 20% to 80% on WikiText-2 and six classification datasets. As shown in Figure 7, MTL-SVD consistently outperforms all baselines, and the performance gain compared to the best-performing baseline enlarges with the increase of compression ratios.

5.2 Inference Speedup of MTL-SVD

To illustrate the inference speedup achieved by MTL-SVD, we measure the numbers of tokens that the original LLaMA-7B and its compressed version by MTL-SVD under different compression ratios generate per second on a single NVIDIA A100 GPU. For a fair measurement, we fixed the batch size to 4, the prefill length to 1024, and the decoding length to 256. As shown in Figure 8, MTL-SVD consistently ensures an enhancement in the generation speed across all the compression ratios. More importantly, the enhancement becomes more significant as the compression ratio increases, resulting in a speedup of 1.29x under 20% compression ratio, 1.63x under 40% compression ratio, 2.08x under 60% compression ratio, and 2.71x on GPU under 80% compression ratio.

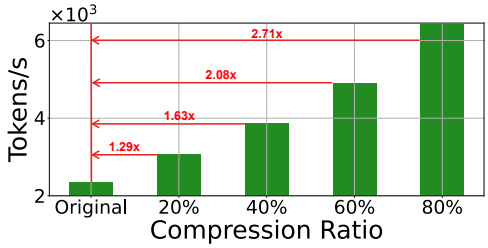


Figure 8: Throughput (Tokens/s) achieved by original LLaMA-7B and its compressed version by MTL-SVD under different compression ratios on a single NVIDIA A100 GPU. We fixed the batch size to 4, prefill length to 1024, and decoding length to 256. The speedup compared to the original LLM is marked in red.

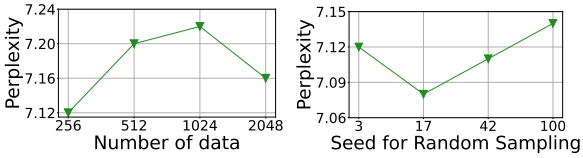
Table 3: Perplexity (↓) of compressed LLaMA-7B on WikiText-2 by SVD-LLM and MTL-SVD with individual / both components under 20% compression ratios.

SVD-LLM	MTL-SVD (A)	MTL-SVD (T)	MTL-SVD (A+T)
7.94	7.91 (↓1%)	7.43 (↓6%)	7.12 (↓10%)

5.3 Ablation Study

MTL-SVD consists of two key components, both of which aim to minimize the truncation loss. In our ablation study, we first evaluate the individual contribution of the two components to the compression performance. Since both components fully utilized the whitening matrix S, which is calculated with a randomly selected calibration set, we next evaluate the changes of the characteristics of the calibration set to the performance of MTL-SVD.

Modular Sensitivity Study: We first evaluate the separate contributions of the two key components (i.e., dynamic compression ratio allocation and loss-minimized weight truncation) of MTL-SVD. Let MTL-SVD (A) denote the version of MTL-SVD with dynamic compression ratio allocation only; MTL-SVD (T) denote the version of MTL-SVD with loss-minimized weight truncation only; and MTL-SVD (A+T) denote the version of MTL-SVD with both dynamic compression ratio allocation and loss-minimized weight truncation. The results are shown in Table 3. We have two observations. (1) MTL-SVD (A), MTL-SVD (T) and MTL-SVD (A+T) outperform MTL-SVD, highlighting the individual effectiveness of the two modules in achieving good compression performance. (2) MTL-SVD (A+T) consistently outperforms MTL-SVD (A) and MTL-SVD (T), highlighting the effectiveness of combining the two components for further enhancing the performance in MTL-SVD.



(a) Various Data Number (b) Various Samping Seed

Figure 9: Perplexity of LLaMA-7B under 20% compression ratio using calibration data sampled with different number or seeds from WikiText-2.

Impact of Calibration Set: We then examine the impact of calibration set on the compression performance of MTL-SVD. Specifically, we measure the changes of perplexity of LLaMA-7B compressed by MTL-SVD under 20% compression ratio on WikiText-2 when using the default calibration set but with various number and sampling seed. As shown in Figure 9, the changes of both data number and sampling seed on calibration set incur no more than 1% fluctuation to the final performance, demonstrating that MTL-SVD is not sensitive to the calibration set.

5.4 Comparison with Other Types of Post-training LLM Compression Methods

MTL-SVD is orthogonal to other post-training LLM compression methods, including quantization and pruning. In this experiment, we compare the performance of MTL-SVD with the state-of-the-art structured pruning-based and quantization-based LLM compression methods.

Table 4: Perplexity (↓) of LLaMA-7B compressed by structured pruning methods and MTL-SVD under various weight memory budget on WikiText-2.

	Perplexity under weight memory budget			
Method	10 GB	9 GB	8 GB	7 GB
LLM-Pruner	9.88	12.21	18.94	21.68
SliceGPT	8.78	12.73	16.39	27.41
BlockPruner	9.40	12.76	19.78	43.05
MTL-SVD	**7.84** (↓17%)	**8.48** (↓34%)	**10.17** (↓49%)	**15.62** (↓28%)

Comparison with Structured Pruning. First, we compare MTL-SVD with three state-of-the-art post-training structured pruning-based LLM compression methods: LLM-Pruner (Ma et al., 2023), SliceGPT (Ashkboos et al., 2024), and Block-Pruner (Zhong et al., 2024) under the same compressed weight memory, ranging from 10GB to 7GB. The results are shown in Table 4. MTL-SVD outperforms all state-of-the-art structured pruning-based LLM compression methods. In particular, MTL-SVD achieves up to 28% reduction in perplexity under the 7G memory budget.

Comparison with Quantization. Finally, we compare MTL-SVD with two state-of-the-art post-training quantization-based LLM compression methods that push the frontier to 1-bit quantization: BiLLM (Huang et al., 2024), and PB-LLM (Yuan et al., 2024). Since MTL-SVD is orthogonal and easy to implement together with quantization-based methods, we compare both MTL-SVD in 16-bit format and MTL-SVD in 2-bit format quantized by the state-of-the-art 2-bit post-training quantization methods QuIP# (Tseng et al., 2024). The results on LLaMA-7B are shown in Table 5: We have two observations: (1) MTL-SVD-16 bit achieves a performance comparable to several 1-bit post-training quantization methods. In particular, it achieves 5% lower perplexity than PB-LLM (Yuan et al., 2024), highlighting the effectiveness of MTL-SVD in compressing LLM to the extremely low memory budge similiar to 1-bit quantization. (2) MTL-SVD-2 bit outperforms the state-of-the-art 1-bit post-training quantization methods. In particular, it even achieves 69% lower perplexity that the best-performing baselines, providing a new research avenue that combining MTL-SVD with recent standard quantization methods to further enhancing the compressing performance.

Table 5: Perplexity (↓) of LLaMA-7B compressed by 1-bit quantization methods and MTL-SVD with LoRA fine-tuning on WikiText-2.

Method	Data Type	Weight Memory	Perplexity
PB-LLM	1-bit	1.9 GB	104.83
BiLLM	1-bit	1.5 GB	47.67
MTL-SVD	16-bit	1.5 GB	99.64
MTL-SVD	2-bit	1.5 GB	**14.73** (↓69%)

6 Conclusion

In this paper, we presented MTL-SVD, a SVD-based post-training LLM compression method. MTL-SVD addresses the limitation of existing methods that conducting SVD truncation incur huge truncation loss. To reduce the truncation loss, MTL-SVD first employs a dynamic compression ratio allocation strategy to effectively balance truncation loss across different weight matrices in the LLM. It further introduces a loss-minimized weight truncation to ensure that the truncated singular values result in the lowest possible truncation loss. Our evaluation results demonstrate the superiority of MTL-SVD over state-of-the-arts.

7 Limitations

Although MTL-SVD outperforms existing SVD-based LLM compression methods, its capabilities can still get improved a lot. For example, under 90% compression ratio, there is still a small performance gap when compared with the state-of-the-art quantization methods. We aim to fill this gap in our future work.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *NAACL-HLT (1)*, pages 2357–2367. Association for Computational Linguistics.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *ICLR*. OpenReview.net.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439. AAAI Press.

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. *CoRR*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

G.H. Golub, Alan Hoffman, and G.W. Stewart. 1987. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its Applications*, 88-89:317–327.

Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchán. 2023. A survey of generative AI applications. *CoRR*, abs/2306.02781.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *ICLR*. OpenReview.net.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and

Xiaojuan Qi. 2024. Billm: Pushing the limit of post-training quantization for llms. In *ICML*. OpenReview.net.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL (1)*, pages 3214–3252. Association for Computational Linguistics.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4A8KV4 quantization and system co-design for efficient LLM serving. *CoRR*, abs/2405.04532.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR (Poster)*. OpenReview.net.

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 *EMNLP*, pages 2381–2391. Association for Computational Linguistics.

Stephen G. Nash and Jorge Nocedal. 1991. A numerical study of the limited memory bfgs method and the truncated-newton method for large scale optimization. *SIAM J. Optim.*, 1:358–372.

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.*, 21:140:1–140:67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*, pages 8732–8740. AAAI Press.

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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *ICML*. OpenReview.net.

Zhongwei Wan, Xin Wang, et al. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*.

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. *arXiv preprint arXiv:2401.01923*.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024b. SVD-LLM: truncation-aware singular value decomposition for large language model compression. *CoRR*, abs/2403.07378.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. 2024. PB-LLM: partially binarized large language models. In *ICLR*. OpenReview.net.

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. *CoRR*, abs/2312.05821.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*, pages 4791–4800. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2024. Blockpruner: Fine-grained pruning for large language models. *CoRR*, abs/2406.10594.

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. *Preprint*, arXiv:2404.14294.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *CoRR*, abs/2308.07633.

A Appendix

A.1 Compression Speed Analysis

Apart from measuring the compressed LLMs' performance, we also evaluate the compression speed of MTL-SVD and the baselines, including ASVD and SVD-LLM. Specifically, we measured the A100 GPU hours used for MTL-SVD and the baseline methods when compressing LLaMA-7B under 20% compression ratio. The results show that ASVD takes about 5.5 GPU hours, SVD-LLM takes about 15 minutes, while MTL-SVD takes about 1.5 GPU hours to run the compression. Since MTL-SVD not only requires computing the whitening matrices, which consumes the majority of time in SVD-LLM, but also needs to run its own key modules during compression, it is expected to consume longer time than SVD-LLM. However, it is still faster than ASVD, indicating that its compression time is still acceptable for post-training LLM compression. Moreover, our time breakdown analysis shows that the dynamic compression ratio allocation only requires 2 minutes, while the loss-minimized weight truncation takes up almost all 1.5 hours. Although this module is time-consuming, it deserves the time cost, as it helps MTL-SVD significantly reduce its truncation loss during the SVD compression.

A.2 Optimization Details

In our experiments, we use the L-BFGS (Nash and Nocedal, 1991) algorithm to perform optimization in loss-minimized weight truncation due to its benefits for fast convergence, especially when dealing with large-scale problems. We set the learning rate to 0.01 and the maximum iteration to 40. Our empirical results show that the loss drops rapidly and often converges in 20 iterations. Therefore, the time consumed to perform this optimization is still acceptable. As mentioned in Appendix A.1, it only takes about 1.5 hours to optimize all weight matrices in LLaMA-7B, which also meets our initial goal of designing an efficient and practical way to minimize truncation loss.