
MoE-SVD: Structured Mixture-of-Experts LLMs Compression via Singular Value Decomposition

Wei Li^{*1} Lujun Li^{*2} Hao Gu² You-Liang Huang³ Mark Lee¹ Shengjie Sun⁴ Wei Xue² Yike Guo²

Abstract

Mixture of Experts (MoE) architecture improves Large Language Models (LLMs) with better scaling, but its higher parameter counts and memory demands create challenges for deployment. In this paper, we present MoE-SVD, a new decomposition-based compression framework tailored for MoE LLMs without any extra training. By harnessing the power of Singular Value Decomposition (SVD), MoE-SVD addresses the critical issues of decomposition collapse and matrix redundancy in MoE architectures. Specifically, we first decompose experts into compact low-rank matrices, resulting in accelerated inference and memory optimization. In particular, we propose selective decomposition strategy by measuring sensitivity metrics based on weight singular values and activation statistics to automatically identify decomposable expert layers. Then, we share a single V-matrix across all experts and employ a top-k selection for U-matrices. This low-rank matrix sharing and trimming scheme allows for significant parameter reduction while preserving diversity among experts. Comprehensive experiments on Mixtral, Phi-3.5, DeepSeek, and Qwen2 MoE LLMs show MoE-SVD outperforms other compression methods, achieving a 60% compression ratio and 1.5x faster inference with minimal performance loss.

1. Introduction

Mixture of Experts (MoE) (Cai et al., 2024b) have demonstrated promising advancements in the realm of recent large

^{*}Equal contribution ¹University of Birmingham ²The Hong Kong University of Science and Technology ³The Hong Kong University of Science and Technology (Guangzhou) ⁴AISpeech Co., Ltd.. Correspondence to: Mark Lee <M.G.Lee@bham.ac.uk>, Yike Guo <yikeguo@ust.hk>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

language models (LLMs) (*e.g.*, MiniMax-01 (Minimax et al., 2025), Qwen2.5-Max (Team, 2024) and DeepSeek-V3 (DeepSeek-AI et al., 2024)). These architectures incorporate multiple experts and employ a sparse gating mechanism, enabling efficient computation and facilitating the scaling of LLMs within the constraints of limited computational resources (Dai et al., 2024). Despite these advantages, MoE LLMs still face several challenges: **(1) Immense Parameter Sizes:** MoE models generally have a larger number of parameters than dense models (Gu et al., 2025), which can make them difficult to train and deploy, especially in resource-constrained environments. **(2) Memory Overhead:** MoE models can suffer from memory inefficiency due to the need to store and access multiple expert weights and biases, potentially hindering their deployment on devices with limited memory (Song et al., 2023).

Limitations of Traditional MoE Compressions: To address the challenges of large parameter size and memory overhead, some MoE-specific compression methods have been proposed to prune unimportant experts or weights. For example, NAAE (Lu et al., 2024) proposes task-specific expert pruning and dynamic skipping, while MoE-Compression (He et al., 2024) evaluates various types of sparse schemes across multiple MoE components. Although these techniques show promise, they suffer from certain limitations: **(1) Performance Degradation**, especially under high compression ratios, often necessitating costly and time-consuming retraining. For instance, pruning 25% of experts in Mixtral-8×7B results in a 23% performance drop (He et al., 2024). **(2) Hardware Dependency:** Some semi-structured sparse methods only gain speedup on NVIDIA Ampere and Hopper architecture GPUs, limiting their general applicability. **(3) Limited Acceleration:** Some MoE compression methods only reduce the number of experts without significantly reducing the size of the activated experts (Liu et al., 2024a), resulting in minimal speedup during inference. MoE-Compression (He et al., 2024) report that eliminating 12.5% of experts yields less than a 1% speed boost. In contrast, recent Singular Value Decomposition (SVD) techniques (Hsu et al., 2022) are hardware-independent and successfully compact LLM to high compression ratios without additional training. These facts encourage us to explore SVD as an alternative to pruning.

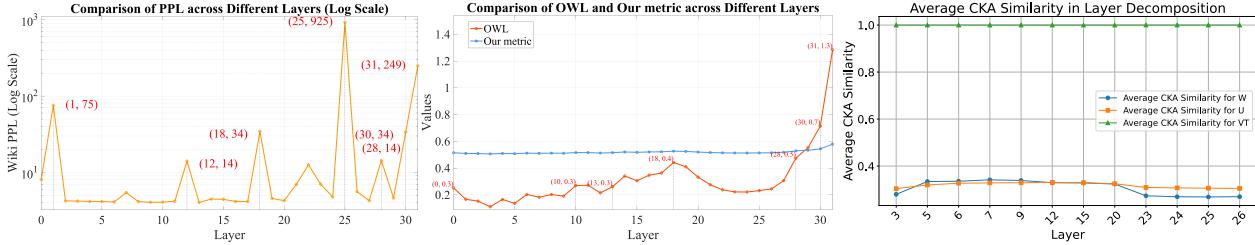


Figure 1: Perplexity of 50% per-layer SVD decomposition (*left*), per-layer values of OWL & our metric (*middle*), mean CKA similarity (Kornblith et al., 2019) of decomposed V & U matrix and original matrix of each expert layer (*right*). These results are obtained for Mixtral-8x7B on WikiText-2.

However, we directly apply these SVD-based methods to compress MoE models, resulting in serious performance collapse. For example, Mixtral-8x7B with ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2024) reach over 1000 perplexity on WikiText-2. This naturally raises key questions: ***Why SVD-based methods fail on MoE LLMs and how to solve this?***

Our New Observations: To answer these questions, we individually decompose each expert layer in Figure 1 and uncover observations: **(1) Decomposition Sensitivity:** Some expert layers are more sensitive to SVD decomposition than others. For example, initial and final layers of decomposition can lead to drastic performance loss. This indicates that layer-wise non-uniform decomposition is important for MoE LLMs. **(2) Model Statistic Disparities:** Activation outliers in OWL (Yin et al., 2023) are an effective pre-layer importance statistic and metric for dense LLM. However, we notice that their values on MoE in Figure 1 (*middle*) do not match the pre-layer decomposition result in (1). This could be attributed to biases derived from multi-expert design and dynamic activation in MoE LLMs. **(3) Expert Redundancy:** Expert merging methods (Liu et al., 2024a) show various experts are similar in the weight space and contain significant redundancy. Our Figure 1 (*right*) indicates high similarity of decomposed V-matrices, which can further share weights or trim redundant matrices.

"Different problems require different solutions."

— Albert Einstein

Our Novel Framework: As this well-said quote goes, the sparse-activated MoE dynamic architecture differs from common LLMs and deserves customized decomposition schemes based on the above observations. To this end, we develop MoE-SVD, a novel compression framework specifically designed for MoE LLMs. Our MoE-SVD leverages SVD to decompose expert layers in a structured manner, creating a naturally sparse expert structure that reduces computational costs while maintaining model expressiveness. The core innovation of MoE-SVD lies in: **(1) Selective Decomposition Strategy:** Unlike previous SVD approaches

that apply uniform compression across different layers, our method introduces a sensitivity metric derived from matrix singular values and activation statistics, facilitating adaptive decomposition. As illustrated in Figure 1 (*middle*), this metric accurately identifies the sensitive expert layers, allowing for more targeted compression. **(2) Low-rank Matrix Sharing and Trimming:** To further minimize parameter redundancy, we introduce V-matrix sharing, where the most frequently sampled V-matrix is retained and shared across all experts. In addition, we apply U-matrix trimming by selecting the top-k U-matrices based on sampling frequency, while discarding the less frequently used matrices. This strategy significantly minimizes the number of parameters, while ensuring the diversity for effective MoE functioning. With these innovative schemes, our MoE-SVD offers substantial parameter reduction, creates a naturally sparse expert structure for faster inference, and can be deployed on standard computing infrastructure without requiring additional training phases. This flexible framework allows high compression ratios and strikes an optimal balance between computational efficiency and model performance.

Validation and Results: Extensive experiments demonstrate Our MoE-SVD method achieves state-of-the-art performance in compressing MoE models while maintaining their performance on various language modeling and common sense reasoning datasets. The results show that MoE-SVD outperforms other methods such as SVD, ASVD, and SVD-LLM across all compression ratios from 20% to 60%, on both Mixtral-8x7B and Phi-3.5-MoE models. For example, on the Mixtral-8x7B model, MoE-SVD achieves a 20% compression ratio with only a 2% drop in performance, while the other methods experience a significant drop in performance. Similarly, on the Phi-3.5-MoE, MoE-SVD achieves a 40% compression ratio with only a 5% drop in performance. These results demonstrate the effectiveness of MoE-SVD in compressing MoE models while maintaining their performance. In addition, our MoE-SVD can generalize well to other MoE LLMs such as DeepSeekMoE-16B and Mixtral-8x22B, and can be further improved in performance by LoRA fine-tuning and efficiency with quantiza-

Table 1: Comparison of our MoE-SVD to general LLM compressors (*e.g.*, Wanda, SparseGPT, LoSparse) and MoE-specific compressors (*e.g.*, MC-SMoE, MoE- I^2 , MoE-Compression).

Method	Pipeline	Decomposition	Training	MoE-specific Strategy	Evaluated Models
Wanda (2023)	Pruning without weight updates	None	None	None	LLaMA, LLaMA-2 (7B, 13B, 70B)
SparseGPT (2023)	Pruning and weight updates	None	ADMM-tuning	None	OPT-175B, BLOOM-176B
LoSparse (2023b)	Decompose→sparse	Low-rank + sparse	Fine-tuning	None	BERT-base (110M), DeBERTaV3-large (131M)
MC-SMoE (2024g)	Merge→compress	Low-rank + sparse	Fine-tuning (distill)	Frequency-based merging	T5-base (220M), Switch-base-32 (2.0B)
MoE- I^2 (2024)	Search→expert drop→decompose	Low-rank + sparse	Fine-tuning	Genetic Search, KT-Reception Field	Qwen, Mixtral-8x7B
MoE-Compression (2024)	Pruning and layer dropping	None	None	Layer-dropping, sparsity	Mixtral-8x7B, DeepSeekMoE
MoE-SVD (Ours)	Decompose→matrix share-trim	Low-rank	None	Selective SVD, matrix share-trim	Mixtral-8x7B/22B, Phi-3.5-42B, DeepSeek!QwenMoE

tion. We summarize our contribution as follows:

- To overcome limitations of existing methods, we open new doors for MoE compression from the SVD technical route. We derive series of important findings about decomposition collapse, statistic discrepancies, and redundancy, providing insights into this new area.
- We introduce MoE-SVD, new structured compressor for MoE LLMs. Our MoE-SVD enjoys benefits: high compression ratios, clear inference acceleration, free from specialized hardware and extra training
- We propose a selective decomposition that adaptively applies SVD and develop low-rank matrix sharing and trimming techniques. By sharing V-matrices across experts and trimming redundant U-matrices, we achieve significant parameter reduction while maintaining expert diversity and model performance.
- Extensive experiments demonstrate the effectiveness of MoE-SVD on Mixtral, Phi-3.5 and DeepSeek MoE. Our MoE-SVD consistently outperforms other SVD-based methods across 20% \sim 60% compression ratios and achieves 1.2 \times \sim 1.5 \times inference speedups.

2. Related Work

Mixture of Experts. architecture’s ability to achieve superior scaling laws at reduced costs (Clark et al., 2022) has led to its widespread adoption in recent Large Language Models (LLMs) (Jiang et al., 2024; Dai et al., 2024). Recent advancements in MoE focus on refining expert structures (Dai et al., 2024), enhancing router designs (Zhou et al., 2022), and developing training strategies (Liu et al., 2023). However, MoE LLMs still face challenges, including increased parameter budgets due to expert replication (He et al., 2023), communication costs that enhance latency (Xue et al., 2024b), and significant memory overhead issues (Li et al., 2024g), posing challenges to their efficiency.

MoE Compression. Table 1 highlights the key differences between our method and other compressor: (1) Rather than LLM compressors like Wanda (Sun et al., 2023) (mostly unstructured), we propose new MoE-specific structured compression paradigms. (2) Compared to other expert pruning

methods (Lu et al., 2024; He et al., 2024) requiring fine-tuning and drop inactive experts without acceleration, MoE-SVD primarily exploits SVD to reduce the size of activated experts for acceleration without extensive retraining. (3) Our MoE-SVD enhances the SVD for large-scale MoEs without expert merging and fine-tuning, in contrast to MC-SMoE (Li et al., 2024g), which only addresses small-scale MoE by first merging experts and then applying vanilla decomposition before fine-tuning. (4) MoE-SVD performs low-rank matrix sharing and trimming, avoiding the direct dropping of entire experts like MoE- I^2 (Yang et al., 2024) and MoE-Compression (He et al., 2024), which prevents drastic performance loss. (5) Our method is search-free, avoiding the huge search overhead of in MoE- I^2 (Yang et al., 2024) and EEP (Liu et al., 2024a). (6) Other approaches include post-training quantization (Li et al., 2024f) and system optimization (*e.g.*, expert offloading (Xue et al., 2024a), parallelism (Cai et al., 2024a), and switching (Liu et al., 2024b)). Our MoE-SVD focuses solely on expert compression and remains orthogonal to these methods.

Singular Value Decomposition. Recently, several SVD-based methods have been proposed for compressing LLMs (Golub et al., 1987). FWSVD (Hsu et al., 2022) introduces a weighted low-rank factorization, while ASVD (Yuan et al., 2023) proposes an activation-aware SVD method that considers the activation patterns of the model’s layers to improve compression efficiency. Meanwhile, SVD-LLM (Wang et al., 2024) adopts truncation-aware data whitening and layer-wise parameter update strategies to achieve better compression ratios. In contrast to these general SVD-based methods, our MoE-SVD is specifically designed for MoE LLMs, addressing their unique challenges (*e.g.*, decomposition sensitivity and expert redundancy). Additionally, while these methods typically uniformly decompose every layer in LLMs, our method employs adaptive decomposition across various expert layers in MoE LLMs.

3. Methodology

Our MoE-SVD introduces SVD expert decomposition, selective decomposition strategy, low-rank matrix sharing and trimming to reduce model parameters while maintaining performance. The main process of MoE-SVD is illustrated in Figure 2. More algorithm details are in Appendix C.

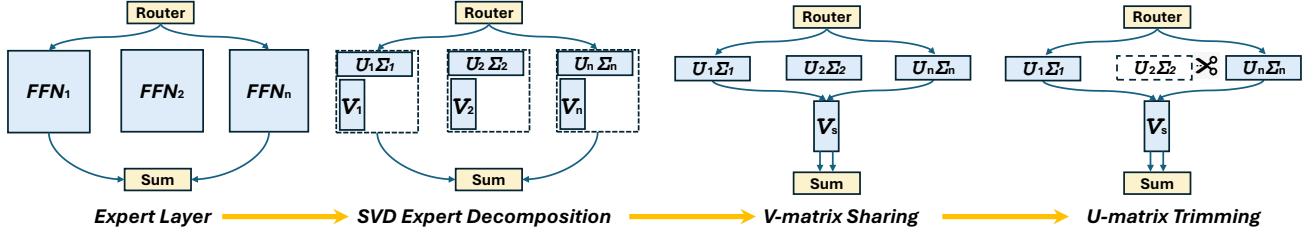


Figure 2: Pipeline of MoE-SVD. We first selectively decompose expert layers with SVD, dividing them into U and V matrices. Then, we present V-matrix sharing and U-matrix trimming steps. For V-matrix sharing, we retain only a single V-matrix V_s and share it across experts. For U-matrix trimming, we perform frequency-based top-k selection of U-matrices and trim out unselected ones.

3.1. SVD Expert Decomposition in MoE LLMs

Recap of MoE Formulation. MoE architectures in LLM enhance model capacity and efficiency by using expert-based Feed-Forward Network (FFN) layers for different input tokens. The output y of the MoE-FFN layer for input x is computed as:

$$y = \sum_{i=1}^N G(x)_i \cdot E_i(x), \quad (1)$$

where N is the number of experts, $G(x)$ is the gating function. The gating function $G(x)$ typically employs a top- k selection mechanism, where only the top- k experts are activated $G'(x) = \text{TopK}(G(x), k)$, resulting in a sparse output. and $E_i(x)$ is the output of the i -th expert. Each expert E_i is a standard FFN with two or three fully-connected layers. Our method and other MoE compressors are focused on compressing FFN experts, which account for the majority of parameters and memory overhead in MoE models.

Activation-weighted SVD for Expert Decomposition. In our SVD-based framework, we apply SVD to decompose the weights of expert layers. To capture outliers and low-rank features in activations, we scale the original matrix W to activation-weighted matrix $W \cdot S$ before SVD, where S is obtained via cholesky decomposition of activation gram matrix (see more details in Appendix D.1). Consider an MoE model with N experts, each fully-connected layer represented by the activation-weighted weight matrix $W_i \in \mathbb{R}^{m \times n}$, where $i \in \{1, \dots, N\}$. We begin by applying activation-weighted SVD to each expert matrix:

$$W_i = U_i \Sigma_i V_i^T, \quad (2)$$

where $U_i \in \mathbb{R}^{m \times m}$ and $V_i \in \mathbb{R}^{n \times n}$ are orthogonal matrices containing the left and right singular vectors, respectively, and $\Sigma_i \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values in descending order, respectively. To create a sparse MoE structure, we first factorize each expert W using the SVD decomposition and then truncates the top- k singular values and their corresponding singular vectors and finally reconstruct an approximated weight matrix:

$$\{E_i\}_{i=1}^k = \{u_i \cdot \sigma_i \cdot v_i^T\}_{i=1}^k, \quad (3)$$

where u_i and v_i are the i -th columns of U and V respectively, and σ_i is the i -th singular value. Our SVD-based

expert decomposition creates a naturally sparse expert structure, potentially reducing computational costs. The number of experts can be easily adjusted, allowing for fine-grained control over the MoE’s capacity and computational requirements, avenues for compress MoE LLMs.

3.2. Selective Decomposition Strategy

To determine the sensitivity of expert layers in the MoE to decomposition, we employ a selective decomposition strategy. This approach is based on a carefully crafted sensitivity metric that considers both the singular value decomposition of expert weight matrices and the activation patterns of these experts during inference. For a layer with N experts, we normalize these sensitivities using expert sampling frequency to obtain the layer-wise sensitivity metric S_L :

$$S_L = \sum_{i=1}^N f_i \cdot p_i \cdot a_i, \quad (4)$$

where f_i represents the sampling frequency of the i -th expert during router selection, p_i denotes the principal rank (number of large value components) of singular vectors $\Sigma_i = \text{diag}(\sigma_{i,1}, \sigma_{i,2}, \dots, \sigma_{i,d})$ obtained from the SVD of the i -th expert’s weight matrix, and a_i measures the proportion of activation outliers of the i -th expert exceeding the mean absolute activation value (see more details in Appendix D.1). To apply selective decomposition, we set a threshold τ based on the desired compression ratio. Expert layer with sensitivity $S_i \geq \tau$ are preserved without decomposition, while those below the threshold undergo SVD decomposition. This process is repeated for each layer in the network. This selective decomposition strategy allows for a nuanced approach to MoE compression, preserving the most important experts while reducing the computational footprint of less critical components.

Decomposable Expert Layer Analysis. To better understand our selective decomposition, we show layer decomposition results for Mixtral-8×7B and Phi-3.5-MoE in Figure 3. As the compression increases from 20% to 60%, both models exhibit a gradual increase in decomposed layers, albeit with distinct characteristics. Mixtral-8×7B displays a more

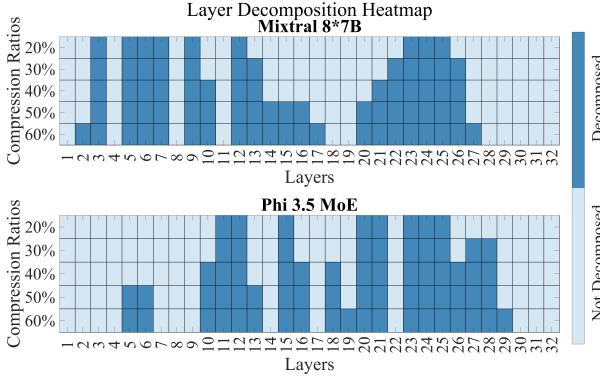


Figure 3: Decomposed expert layers in MoE-SVD.

aggressive decomposition pattern, with approximately half of its layers decomposed at 60% compression, whereas Phi-3.5-MoE demonstrates greater resilience, maintaining more undecomposed layers at higher compression ratios. Notably, both models consistently undecompose their initial and final layers across all compression levels, suggesting the critical nature of these layers for maintaining model performance. In contrast, certain middle layers in both architectures show remarkable tendencies to decomposition. Mixtral-8x7B exhibits a block-like decomposition pattern, while Phi-3.5-MoE showcases a more uniform distribution of decomposed layers. These observations reveal intriguing patterns and present insights for our selective decomposition of expert layers under layer-wise MoE compression.

3.3. Low-rank Matrix Sharing and Trimming

Motivation: For MoE models, different expert matrices contain some similarities and can be merged (Liu et al., 2024a). As shown in Figure 1 (*right*), decomposed V-matrices share certain similarities. Consider two expert matrices W_1 and W_2 with SVD decompositions $W_1 = U_1 \Sigma_1 V_1^T$ and $W_2 = U_2 \Sigma_2 V_2^T$. The similarity in their output transformations can be quantified by the Frobenius inner product of their V-matrices $\langle V_1, V_2 \rangle_F = \text{tr}(V_1^T V_2)$. For experts trained on similar tasks, this inner product is often close to high values, indicating high similarity in output transformations. Consequently, we can perform a more fine-grained matrix selection and sharing, achieving a superior trade-off between performance and the number of parameters.

V-matrix Sharing: We compress MoE by retaining only the V-matrix with the highest router sampling frequency and sharing this matrix across all experts. This method significantly reduces the model’s memory footprint while preserving crucial directional information in the feature space. The router sampling frequency $f(V_i)$ for each expert i is computed based on the routing decisions made by the gating network $G(x)$. The shared V-matrix, denoted as V_s ,

is selected as follows:

$$V_s = \arg \max_{V_i} f(V_i), \quad (5)$$

$$f(V_i) = \frac{\sum_{x \in \mathcal{X}} \mathbb{I}[i \in \text{TopK}(G(x), k)]}{|\mathcal{X}|},$$

where \mathcal{X} represents the set of all input tokens, $\mathbb{I}[\cdot]$ is the indicator function, and k denotes the number of experts selected by the top-k gating mechanism. After selecting V_s , we update all expert matrices to use this shared V-matrix:

$$E_i \approx U_i \Sigma_i V_s^T, \quad i = 1, \dots, N, \quad (6)$$

The shared V_s matrix encapsulates common output space transformations across all experts. For the expert matrix W_i using the shared V-matrix V_s , its expected reconstruction error is $\mathbb{E}[\|W_i - \tilde{W}_i\|_F^2] = \mathbb{E}[\|W_i - U_i \Sigma_i V_s^T\|_F^2]$. Minimizing this error is equivalent to maximizing the correlation between W_i and $U_i \Sigma_i V_s^T$. Given that V_s is chosen based on the highest router sampling frequency, it represents the most commonly used output transformation. Thus, sharing V_s minimizes the expected reconstruction error across all experts.

U-matrix Trimming: For the remaining U-matrices, we employ a top-k selection strategy based on router sampling frequency. The diversity among experts is primarily maintained through the unique $U_i \Sigma_i$ components. Typically, we set $k = 2$ to balance parameter efficiency and expert diversity. The selected U-matrices for each expert are determined as follows:

$$\{U_{i,1}, U_{i,2}\} = \text{TopK}(\{U_j \mid f(V_j) > f(V_i)\}, k = 2), \quad (7)$$

where TopK selects the k U-matrices with the highest router sampling frequencies among those experts more frequently sampled than expert i . The final expert function for expert i becomes:

$$E_i(x) = (U_{i,1} \Sigma_{i,1} + U_{i,2} \Sigma_{i,2}) V_s^T x. \quad (8)$$

where $\Sigma_{i,1}$ and $\Sigma_{i,2}$ are the corresponding singular value matrices for the selected U-matrices.

Parameter Reduction: Our method achieves significant parameter reduction compared to the original MoE model with $N \times m \times n$ parameters. After applying our low-rank decomposition, each expert matrix W_i is decomposed into $U_i \Sigma_i \in \mathbb{R}^{m \times r}$, $\Sigma_i \in \mathbb{R}^{r \times r}$, and $V_i \in \mathbb{R}^{r \times n}$, resulting in $N \times (m \times r + r \times n)$ parameters per expert. With V-matrix sharing, we retain only one $V_s \in \mathbb{R}^{r \times n}$ matrix shared across all experts, reducing the parameter count by a factor of N for the V-matrices. Furthermore, with U-matrix selection, we typically select $k = 2$ U-matrices, reducing the parameter count by a factor of $\frac{N}{2}$ for the U-matrices. Thus, the total number of parameters in our compressed MoE model is $m \times r \times 2 + r \times n = 2mr + rn$. Comparing this to the original $N \times m \times n$ parameters, we achieve a

Table 2: Results of MoE-SVD and MoE-SVD \dagger (with LoRA fine-tuning) on three language modeling datasets (measured by perplexity (\downarrow)) and seven common sense reasoning datasets (measured by both individual and average accuracy (\uparrow)). Ratio means model parameter size reduction ratio. Runtime denotes runtime throughput (Tokens/sec) on a single H800 GPU.

Ratio	Method	Runtime	WikiText-2 \downarrow	PTB \downarrow	C4 \downarrow	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average \uparrow
Mixtral-8x7B													
0%	Original	87.7	3.98	12.99	6.78	0.36	0.84	0.76	0.65	0.57	0.82	0.43	0.63
20%	Wanda (2023) (2:4)	91.1 (1.04 \times)	4.72	18.8	8.43	0.32	0.76	0.72	0.55	0.47	0.79	0.36	0.57
	SparseGPT (2023) (2:4)	93.1 (1.06 \times)	4.61	21.11	8.19	0.3	0.77	0.74	0.56	0.45	0.77	0.35	0.56
	LoSparse (2023b)	92.9 (1.05 \times)	953.51	805.16	1273.12	0.2	0.27	0.49	0.28	0.26	0.53	0.20	0.32
	ASVD (2023)	100.8 (1.1 \times)	9.44	47.29	20.30	0.25	0.71	0.66	0.48	0.40	0.73	0.35	0.51
	SVD-LLM (2024)	102.6 (1.2 \times)	13.45	42.72	17.36	0.22	0.62	0.58	0.42	0.29	0.71	0.26	0.44
	MoE-L ² (2024)	102.3 (1.2 \times)	18.80	80.04	23.92	0.24	0.57	0.58	0.43	0.33	0.68	0.23	0.44
	MC-SMoE (2024g)	95.3 (1.09 \times)	1341.36	1316.52	1478.13	0.26	0.28	0.51	0.29	0.25	0.54	0.19	0.33
40%	MoE-Compression (2024)	99.1 (1.13 \times)	6.12	14.67	11.61	0.3	0.73	0.70	0.54	0.46	0.73	0.33	0.54
	MoE-SVD (Ours)	104.7 (1.2 \times)	4.86	19.42	8.98	0.33	0.79	0.74	0.56	0.49	0.78	0.37	0.58
	MoE-SVD\dagger (Ours)	104.7 (1.2 \times)	4.31	14.94	7.82	0.33	0.80	0.73	0.61	0.55	0.81	0.38	0.60
50%	ASVD (2023)	107.8 (1.2 \times)	30.57	196.02	87.74	0.18	0.41	0.58	0.34	0.22	0.59	0.22	0.36
	SVD-LLM (2024)	108.2 (1.2 \times)	254.76	252.25	79.40	0.16	0.43	0.52	0.33	0.22	0.63	0.23	0.36
	MoE-SVD	109.8 (1.3 \times)	6.74	27.73	12.41	0.27	0.72	0.67	0.43	0.38	0.71	0.32	0.50
60%	ASVD (2023)	120.9 (1.4 \times)	86.61	402.60	164.57	0.15	0.42	0.53	0.30	0.22	0.61	0.22	0.35
	SVD-LLM (2024)	122.0 (1.4 \times)	1325.51	1856.62	439.20	0.14	0.33	0.48	0.28	0.21	0.56	0.23	0.32
	MoE-SVD (Ours)	123.9 (1.4 \times)	9.03	42.93	16.18	0.20	0.61	0.63	0.37	0.30	0.63	0.27	0.43
70%	ASVD (2023)	150.2 (1.7 \times)	12524.91	14702.02	11691.72	0.13	0.26	0.51	0.26	0.21	0.53	0.21	0.30
	SVD-LLM (2024)	148.5 (1.7 \times)	10181.25	9284.95	10987.80	0.14	0.26	0.51	0.26	0.22	0.54	0.21	0.30
	MoE-SVD (Ours)	156.1 (1.8 \times)	13.52	130.26	39.54	0.19	0.45	0.55	0.33	0.23	0.62	0.25	0.37
Phi-3.5-MoE													
0%	Original	98.2	3.48	8.43	8.22	0.40	0.77	0.76	0.68	0.56	0.79	0.38	0.62
20%	ASVD (2023)	105.1 (1.1 \times)	7.22	10.66	9.58	0.35	0.73	0.72	0.57	0.49	0.75	0.34	0.56
	SVD-LLM (2024)	104.5 (1.1 \times)	8.34	14.77	12.89	0.31	0.67	0.66	0.53	0.45	0.72	0.22	0.51
40%	MoE-SVD (Ours)	108.6 (1.1 \times)	4.58	10.13	9.93	0.39	0.77	0.73	0.63	0.53	0.78	0.35	0.60
	MoE-SVD\dagger (Ours)	108.6 (1.1 \times)	4.29	10.99	8.81	0.39	0.81	0.74	0.65	0.54	0.79	0.36	0.61
50%	ASVD (2023)	121.5 (1.3 \times)	14.51	22.14	21.82	0.30	0.69	0.63	0.41	0.40	0.68	0.25	0.48
	SVD-LLM (2024)	122.7 (1.3 \times)	38.83	68.52	43.81	0.23	0.56	0.60	0.39	0.31	0.66	0.24	0.43
	MoE-SVD (Ours)	124.8 (1.3 \times)	5.54	11.73	11.94	0.35	0.72	0.72	0.58	0.48	0.75	0.31	0.56
60%	ASVD (2023)	130.4 (1.3 \times)	20.58	33.53	30.26	0.23	0.62	0.62	0.36	0.32	0.66	0.24	0.44
	SVD-LLM (2024)	129.6 (1.3 \times)	6494.87	6451.79	9348.47	0.16	0.29	0.49	0.27	0.23	0.53	0.20	0.31
	MoE-SVD (Ours)	137.2 (1.4 \times)	6.24	13.99	14.99	0.34	0.68	0.69	0.53	0.47	0.72	0.30	0.53
70%	ASVD (2023)	138.8 (1.4 \times)	107.71	208.69	161.40	0.18	0.40	0.53	0.30	0.24	0.59	0.23	0.35
	SVD-LLM (2024)	140.3 (1.4 \times)	7168.09	7101.49	7119.43	0.15	0.28	0.51	0.26	0.22	0.54	0.21	0.31
80%	MoE-SVD (Ours)	148.7 (1.5 \times)	7.46	20.95	21.85	0.30	0.60	0.68	0.46	0.40	0.71	0.25	0.49

Table 3: Results under 20% compression ratio on language modeling and reasoning datasets.

Models	Runtime	WikiText-2 \downarrow	PTB \downarrow	C4 \downarrow	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average \uparrow
DeepSeekMoE-16B Original	52.53	6.38	9.47	9.82	0.33	0.76	0.71	0.58	0.44	0.79	0.31	0.56
DeepSeekMoE-16B MoE-SVD	62.79 (1.21 \times)	6.92	10.48	11.99	0.31	0.75	0.70	0.53	0.42	0.76	0.31	0.54
Mixtral-8x22B Original	66.34	2.95	10.1	6.14	0.37	0.86	0.80	0.67	0.59	0.83	0.50	0.66
Mixtral-8x22B MoE-SVD	78.18 (1.18 \times)	4.03	15.02	8.73	0.37	0.84	0.79	0.61	0.54	0.81	0.44	0.63
Qwen2-57B-A14B Original	42.70	4.32	11.66	9.23	0.33	0.75	0.74	0.63	0.47	0.8	0.39	0.58
Qwen2-57B-A14B MoE-SVD	53.16 (1.25 \times)	5.41	13.26	11.63	0.30	0.74	0.73	0.61	0.45	0.78	0.35	0.56

substantial reduction in the number of parameters, especially when $r \ll \min(m, n)$. The parameter reduction ratio is approximately $\frac{2mr+rn}{Nmn}$, which can be significant for LLMs with high input and output dimensions.

4. Experiments

In this section, we first compare MoE-SVD against other compressors on Mixtral-8x7B and Phi-3.5-MoE at different compression ratios. Then, we conduct evaluation on diverse MoE LLMs, inference efficiency, ablation studies and extend MoE-SVD with quantization. All experiments are conducted on NVIDIA H800 GPUs.

4.1. Experimental Setups

Models and Datasets. To showcase the versatility of our MoE-SVD method, we assess its effectiveness on Mixtral models (8x7B and 8x22B), Phi-3.5-MoE, and DeepSeek-MoE. Mixtral variations employ 8 experts, achieving remarkable language modeling capabilities. Phi-3.5-MoE excels with 16x3.8 B parameters, while DeepSeek-MoE, with 16 B parameters utilizing fine-grained experts, also exhibits superior performance. We evaluate our method across 10 datasets, encompassing 3 language modeling datasets (WikiText-2 (Merity et al., 2017), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020)), along with 7 common sense reasoning datasets (OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), MathQA (Amini et al., 2019), ARC-e, and ARC-c (Clark et al., 2018)) in a

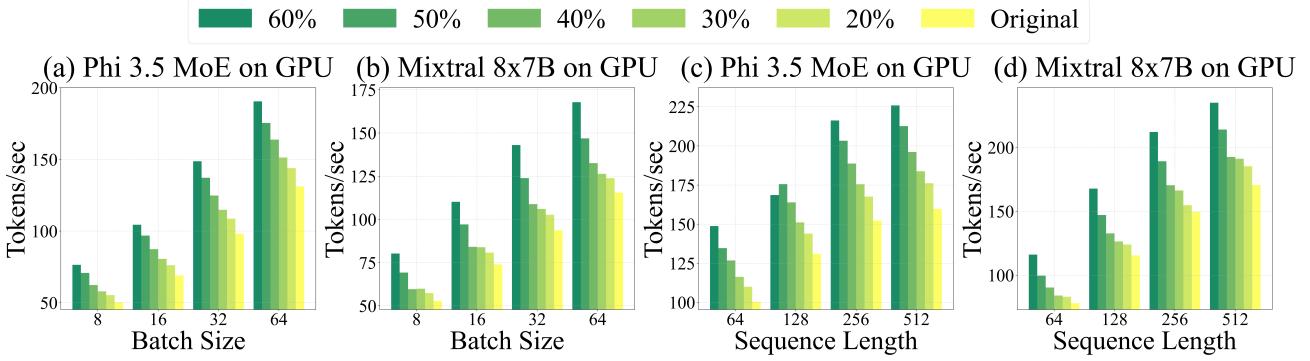


Figure 4: Throughput (Tokens/sec) of Mixtral-8x7B and Phi-3.5-MoE compressed by MoE-SVD at 20%~60% ratios on a single H800 GPU is compared in Figures (a) & (b) for various batch sizes at sequence length = 32, and in Figures (c) & (d) for varying sequence lengths at batch size = 64.

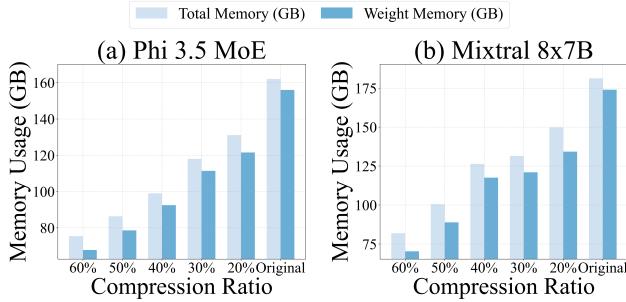


Figure 5: Memory usage (GB) of MoE-SVD at varying compression ratios.

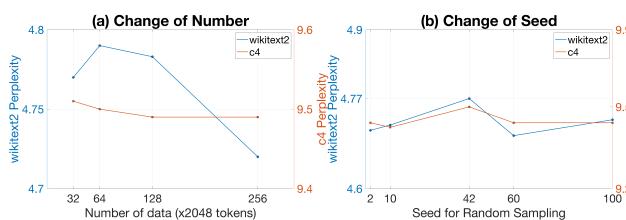


Figure 6: Perplexity of 20% compressed Mixtral-8x22B via calibration data with varying number (a) and seeds (b) from WikiText-2 and C4.

zero-shot setting using the LM-Evaluation-Harness framework (Gao et al., 2023).

Implementation Details. For fair comparisons, we followed the same settings as ASVD and SVD-LLM and used 256 random samples from WikiText-2 as calibration data. We focus on compressing the model without retraining the full model parameters. See Appendix D for more details.

4.2. Performance and Acceleration Results

Performance Comparisons. Results in Table 2 demonstrate that our MoE-SVD method maintains performance close to the original model across various compression ratios, with minimal degradation in perplexity and common-sense reasoning accuracy. For instance, at a 20% compression ratio, MoE-SVD achieves comparable results to the Original, particularly in common-sense reasoning tasks. This highlights the robustness of our method in preserving model performance even under significant compression. Compared to other compression methods, including Wanda, SparseGPT, MC-SMoE, and SVD-based approaches (ASVD, SVD-LLM), MoE-SVD consistently achieves better performance across both language modeling and reasoning datasets. For example, at 20% compression, MoE-SVD outperforms SparseGPT and Wanda in both perplexity and average accuracy. Additionally, MoE-SVD achieves a significant improvement over ASVD and SVD-LLM in all evaluated metrics, with lower perplexity and higher accuracy, particularly in challenging reasoning tasks such as ARC-e and HellaSwag. At a 40% compression ratio, MoE-SVD achieves 0.50 average accuracy of, significantly outperforming SVD-LLM (0.36) and ASVD (0.36), while at a higher 60% compression, MoE-SVD still achieves 0.37 average accuracy, maintaining a clear performance margin compared to SVD-LLM (0.30) and ASVD (0.30). This consistent superiority across varying sparsity levels highlights the robustness and adaptability of MoE-SVD in handling large-scale compression challenges. These results underscore the effectiveness of our method in delivering superior compression performance compared to existing approaches. Furthermore, by leveraging LoRA fine-tuning, MoE-SVD \dagger further improves upon the already strong results of MoE-SVD. At a 20% compression ratio, MoE-SVD \dagger achieves an average accuracy of 0.60, a notable gain over MoE-SVD's 0.58, while also reducing perplexity across language model-

Table 4: Performance of different decomposition settings on Mixtral-8x7B.

Method	WikiText-2↓	PTB↓	C4↓	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average↑
Original	3.98	12.99	6.78	0.36	0.84	0.76	0.65	0.57	0.82	0.43	0.63
SVD (Uniform)	18.80	80.04	23.92	0.24	0.57	0.58	0.43	0.33	0.68	0.23	0.44
SVD (OWL)	16.57	62.13	30.82	0.26	0.61	0.64	0.45	0.32	0.68	0.29	0.46
SVD (Our Selective)	8.67	26.72	12.06	0.24	0.67	0.66	0.48	0.35	0.72	0.28	0.49
MoE-SVD	4.86	19.42	8.98	0.33	0.79	0.74	0.56	0.49	0.78	0.37	0.58

Table 5: Perplexity (↓) of our MoE-SVD with various selective metrics for Mixtral-8x7B on WikiText-2.

Metrics	f_i	p_i	a_i	$f_i \cdot p_i$	$f_i \cdot a_i$	$f_i \cdot p_i \cdot a_i$
Perplexity	12.65	9.85	10.23	9.27	9.31	8.67

Table 6: Perplexity (↓) of our MoE-SVD with various numbers of trimmed matrices for Mixtral-8x7B on WikiText-2.

U trimming	0	1	2	3	4	5	6	7
Perplexity	4.86	5.21	5.98	6.47	7.11	8.67	9.81	10.25

ing datasets. This enhancement demonstrates the potential of fine-tuning techniques to refine and optimize compressed models, further bridging the gap with the original model.

Generalizability of Across Diverse MoE Architectures. To demonstrate the broad applicability of MoE-SVD, we evaluate its performance on two distinct MoE models, DeepSeek-MoE-16B and Mixtral-8x22B, under a 20% compression ratio, as shown in Table 3. The results indicate that compressed models retain competitive performance, with MoE-SVD achieving an 0.54 average accuracy on DeepSeek-MoE-16B and 0.63 on Mixtral-8x22B, compared to 0.56 and 0.66 for their respective original versions. These results highlight the ability of MoE-SVD to preserve a substantial portion of the original model’s reasoning capabilities across diverse datasets and architectures. Similarly, for the Qwen2-57B-A14B model, MoE-SVD achieves 0.56 average accuracy, closely matching the original model. This demonstrates the robustness and generalizability of MoE-SVD across varying MoE architectures, enabling efficient compression without significant loss in performance.

Inference Speed Acceleration and Memory Reduction. Figure 4 shows significant inference speed up for Phi-3.5-MoE and Mixtral-8x7B models as compression ratios increase, with Phi-3.5-MoE achieving up to 1.52× acceleration at 60% compression and Mixtral-8x7B reaching 1.53× at the same compression level, demonstrating consistent gains across batch sizes and sequence lengths. Meanwhile, Figure 9 highlights MoE-SVD’s memory reduction capabilities, with Phi-3.5-MoE’s weight memory decreasing to 67.78 GB (43.45% of original) and Mixtral-8x7B’s to 70.31 GB (40.41% of original) at 60% compression, enabling deployment on memory-limited devices and broadening real-world applicability. These results underscore MoE-SVD’s dual

Table 7: Performance of Mixtral-8x7B with MoE-SVD under 20% compression ratios using calibration data randomly sampled from WikiText-2 (by default in our paper) and C4.

Calibration	WikiText-2↓	PTB↓	C4↓	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Average↑
WikiText-2	4.86	19.42	8.98	0.33	0.79	0.74	0.56	0.49	0.78	0.37	0.58
C4	4.91	19.38	8.70	0.32	0.79	0.73	0.56	0.50	0.79	0.37	0.58

Table 8: Perplexity (↓) of Mixtral 8x7B and Phi-3.5-MoE compressed with GPTQ and MoE-SVD on WikiText-2.

Mixtral 8x7B	GPTQ (4bit)	GPTQ (3bit)	MoE-SVD (4bit)	MoE-SVD (3bit)
Memory	44.5	33.4	35.6	26.7
Perplexity	4.35	6.22	6.93	11.53
Phi-3.5-MoE	GPTQ (4bit)	GPTQ (3bit)	MoE-SVD (4bit)	MoE-SVD (3bit)
Memory	39.0	29.3	27.3	20.5
Perplexity	4.59	6.71	6.64	10.28

benefits of faster inference and reduced memory usage, making MoE LLMs more efficient and accessible.

4.3. Ablation study

Ablation of Selective Decomposition. Table 4 delves into the performance of different selective decomposition methods. Our non-uniform decomposition metric outperforms both uniform SVD and the OWL-based non-uniform SVD method for compressing MoE. Table 5 our metrics can obtain better performance than vanilla a_i , r_i and f_i , respectively, demonstrating the importance of ensemble designs. In addition, our matrix sharing and trimming (MoE-SVD in Table 4) can further reduce the parameter redundancy allowing us to retain more sensitive expert layers, which leads to significant performance gains based on our selective decomposition.

Varying Numbers for Low-rank Matrix Trimming. Our U-matrix trimming serves the same purpose of performance-efficiency tradeoff as other expert pruning. Note that our trimmed U-matrix is derived from decomposable layer, which intrinsically contains more parameter redundancy and contributes slightly smaller to overall performance compared to the other layers. Ablation in Table 6 shows that retaining the Top-2 U-matrix with the highest frequency and trimming the rest 6 U-matrices results in an ideal trade-off and greatly reduces the number of parameters.

Impact of Calibration Data. Table 7 examines the impact of different calibration data sources and results indicate that the choice between WikiText-2 and C4 has minimal impact on the overall performance across various tasks. Figure 6

explores the effects of varying the number of calibration samples and random seed. Results indicate that increasing the number of data samples generally leads to a decrease in perplexity, suggesting improved performance with more samples. Additionally, the choice of random seed shows minimal effect on perplexity across both datasets, demonstrating that our MoE-SVD is relatively robust to sampling variability.

Expanding MoE-SVD via Quantization. The results in Table 8 demonstrate the results of combining MoE-SVD with GPTQ (Frantar et al., 2022) to achieve significant memory savings. Comparing 4-bit and 3-bit quantization levels, our MoE-SVD (4-bit) proves to be on par with direct 3-bit quantization (*e.g.*, GPTQ (3bit)) in terms of both memory efficiency and performance. These findings underscore the effectiveness of the quantization method when combined with MoE-SVD, showcasing its potential for creating memory-efficient models without compromising performance quality.

5. Conclusion

In this paper, we introduce MoE-SVD, a novel SVD-based compression framework tailored for MoE LLMs, effectively streamlining parameter expenses, and memory usage while upholding performance. To combat decomposition collapse stemming from matrix redundancy, we propose innovative solutions, including the selective decomposition strategy and a low-rank matrix sharing and trimming mechanism. The former utilizes a sensitivity metric for automated identification of decomposable layers, while the latter harmonizes parameter efficiency and expert specialization through V-matrix sharing and U-matrix trimming. Our extensive assessments on Mixtral-8×7B and Phi-3.5-MoE models showcase the method’s superiority over existing compressors in preserving model capabilities across diverse tasks. These promising results, encompassing preserved performance, accelerated inference speed, and substantial memory reduction, position MoE-SVD as a significant stride forward in making MoE LLMs more accessible and efficient for real-world applications, paving the way for widespread adoption and deployment of these powerful models.

Limitations: As a common phenomenon, SVD-based methods (*e.g.*, ASVD and SVD-LLM) add communication overhead during tensor parallelism. Nevertheless, our substantial reduction in computation time ultimately optimizes the total computation time, as detailed in the Appendix B.4.

Acknowledgements

This work is funded in part by the HKUST Start-up Fund (R9911), Theme-based Research Scheme grant (T45-205/21-N), the InnoHK funding for Hong Kong Generative AI Research and Development Center, Hong Kong SAR,

and the research funding under HKUST-DXM AI for Finance Joint Laboratory (DXM25EG01).

Impact Statement

This work presents MoE-SVD, a technical framework focused on compressing Mixture of Experts (MoE) models to improve inference speed and reduce memory usage. The research is conducted using open-source benchmarks and does not involve sensitive data, ethical concerns, or societal risks. The primary impact lies in enhancing the efficiency of large language models, enabling their deployment on resource-constrained devices and broadening their accessibility. As such, this work is purely technical and does not pose ethical or societal challenges.

References

- Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *NAACL-HLT (1)*, pp. 2357–2367. Association for Computational Linguistics, 2019.
- Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pp. 7432–7439. AAAI Press, 2020.
- Cai, W., Jiang, J., Qin, L., Cui, J., Kim, S., and Huang, J. Shortcut-connected expert parallelism for accelerating mixture-of-experts. *arXiv preprint arXiv:2404.05019*, 2024a.
- Cai, W., Jiang, J., Wang, F., Tang, J., Kim, S., and Huang, J. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*, 2024b.
- Clark, A., de Las Casas, D., Guy, A., Mensch, A., Paganini, M., Hoffmann, J., Damoc, B., Hechtman, B., Cai, T., Borgeaud, S., et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.
- Dai, D., Deng, C., Zhao, C., Xu, R., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.
- DeepSeek-AI et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Dong, P., Li, L., and Wei, Z. Diswot: Student architecture search for distillation without training. In *CVPR*, 2023a.

- Dong, P., Li, L., Wei, Z., Niu, X., Tian, Z., and Pan, H. Emq: Evolving training-free proxies for automated mixed precision quantization. *arXiv preprint arXiv:2307.10554*, 2023b.
- Dong, P., Li, L., Tang, Z., Liu, X., Pan, X., Wang, Q., and Chu, X. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *ICML*, 2024.
- Dong, P., Li, L., Tang, Z., Liu, X., Wei, Z., Wang, Q., and Chu, X. Parzc: Parametric zero-cost proxies for efficient nas. In *AAAI*, 2025a.
- Dong, P., Li, L., Zhong, Y., Du, D., Fan, R., Chen, Y., Tang, Z., Wang, Q., Xue, W., Guo, Y., et al. Stbllm: Breaking the 1-bit barrier with structured binary llms. In *ICLR*, 2025b.
- Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, 2023.
- Frantar, E., Ashkboos, S., Hoefer, T., and Alistarh, D. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL <https://zenodo.org/records/10256836>.
- Golub, G., Hoffman, A., and Stewart, G. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its Applications*, 88-89:317–327, 1987. ISSN 0024-3795. doi: [https://doi.org/10.1016/0024-3795\(87\)90114-5](https://doi.org/10.1016/0024-3795(87)90114-5). URL <https://www.sciencedirect.com/science/article/pii/0024379587901145>.
- Gu, H., Li, W., Li, L., Qiyuan, Z., Lee, M., Sun, S., Xue, W., and Guo, Y. Delta decompression for moe-based llms compression. In *ICML*, 2025.
- He, S., Ding, L., Dong, D., Liu, B., Yu, F., and Tao, D. PAD-net: An efficient framework for dynamic networks. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14354–14366, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.803. URL <https://aclanthology.org/2023.acl-long.803>.
- He, S., Dong, D., Ding, L., and Li, A. Demystifying the compression of mixture-of-experts through a unified framework. *arXiv preprint arXiv:2406.02500*, 2024.
- Hsu, Y., Hua, T., Chang, S., Lou, Q., Shen, Y., and Jin, H. Language model compression with weighted low-rank factorization. In *ICLR*. OpenReview.net, 2022.
- Huang, W., Liao, Y., Liu, J., He, R., Tan, H., Zhang, S., Li, H., Liu, S., and Qi, X. Mc-moe: Mixture compressor for mixture-of-experts llms gains more. *arXiv preprint arXiv:2410.06270*, 2024.
- Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts, 2024.
- Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.
- Lee, J., Qiao, A., Campos, D. F., Yao, Z., He, Y., et al. Stun: Structured-then-unstructured pruning for scalable moe pruning. *arXiv preprint arXiv:2409.06211*, 2024.
- Li, L. Self-regulated feature learning via teacher-free feature distillation. In *ECCV*, 2022.
- Li, L. and Jin, Z. Shadow knowledge distillation: Bridging offline and online knowledge transfer. In *NeuIPS*, 2022.
- Li, L., Dong, P., Wei, Z., and Yang, Y. Automated knowledge distillation via monte carlo tree search. In *ICCV*, 2023a.
- Li, L., Bao, Y., Dong, P., Yang, C., Li, A., Luo, W., Liu, Q., Xue, W., and Guo, Y. Detkds: Knowledge distillation search for object detectors. In *ICML*, 2024a.
- Li, L., Dong, P., Li, A., Wei, Z., and Yang, Y. Kd-zero: Evolving knowledge distiller for any teacher-student pairs. *NeuIPS*, 2024b.
- Li, L., Peijie, Tang, Z., Liu, X., Wang, Q., Luo, W., Xue, W., Liu, Q., Chu, X., and Guo, Y. Discovering sparsity allocation for layer-wise pruning of large language models. In *NeuIPS*, 2024c.
- Li, L., Sun, H., Li, S., Dong, P., Luo, W., Xue, W., Liu, Q., and Guo, Y. Auto-gas: Automated proxy discovery for training-free generative architecture search. *ECCV*, 2024d.

- Li, L., Wei, Z., Dong, P., Luo, W., Xue, W., Liu, Q., and Guo, Y. Attnzero: efficient attention discovery for vision transformers. In *ECCV*, 2024e.
- Li, P., Jin, X., Cheng, Y., and Chen, T. Examining post-training quantization for mixture-of-experts: A benchmark. *arXiv preprint arXiv:2406.08155*, 2024f.
- Li, P., Zhang, Z., Yadav, P., Sung, Y.-L., Cheng, Y., Bansal, M., and Chen, T. Merge, then compress: Demystify efficient SMoe with hints from its routing policy. In *The Twelfth International Conference on Learning Representations*, 2024g. URL <https://openreview.net/forum?id=eFWG9Cy3WK>.
- Li, W., Li, L., Lee, M., and Sun, S. Als: Adaptive layer sparsity for large language models via activation correlation assessment. In *NeuIPS*, 2024h.
- Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., and Zhao, T. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *ICML*, 2023b.
- Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., and Zhao, T. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20336–20350. PMLR, 23–29 Jul 2023c. URL <https://proceedings.mlr.press/v202/li23ap.html>.
- Liu, B., Ding, L., Shen, L., Peng, K., Cao, Y., Cheng, D., and Tao, D. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer. *arXiv preprint arXiv:2310.09762*, 2023.
- Liu, E., Zhu, J., Lin, Z., Ning, X., Blaschko, M. B., Yan, S., Dai, G., Yang, H., and Wang, Y. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024a.
- Liu, J., Gong, R., Zhang, M., He, Y., Cai, J., and Zhuang, B. Me-switch: A memory-efficient expert switching framework for large language models. *arXiv preprint arXiv:2406.09041*, 2024b.
- Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., Yan, J., and Li, H. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models, 2024.
- Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *ICLR (Poster)*. OpenReview.net, 2017.
- Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*, pp. 2381–2391. Association for Computational Linguistics, 2018.
- MiniMax, Li, A., Gong, B., Yang, B., Shan, B., Liu, C., Zhu, C., Zhang, C., Guo, C., Chen, D., Li, D., Jiao, E., Li, G., Zhang, G., Sun, H., Dong, H., Zhu, J., Zhuang, J., Song, J., Zhu, J., Han, J., Li, J., Xie, J., Xu, J., Yan, J., Zhang, K., Xiao, K., Kang, K., Han, L., Wang, L., Yu, L., Feng, L., Zheng, L., Chai, L., Xing, L., Ju, M., Chi, M., Zhang, M., Huang, P., Niu, P., Li, P., Zhao, P., Yang, Q., Xu, Q., Wang, Q., Wang, Q., Li, Q., Leng, R., Shi, S., Yu, S., Li, S., Zhu, S., Huang, T., Liang, T., Sun, W., Sun, W., Cheng, W., Li, W., Song, X., Su, X., Han, X., Zhang, X., Hou, X., Min, X., Zou, X., Shen, X., Gong, Y., Zhu, Y., Zhou, Y., Zhong, Y., Hu, Y., Fan, Y., Yu, Y., Yang, Y., Li, Y., Huang, Y., Li, Y., Huang, Y., Xu, Y., Mao, Y., Li, Z., Li, Z., Tao, Z., Ying, Z., Cong, Z., Qin, Z., Fan, Z., Yu, Z., Jiang, Z., and Wu, Z. Minimax-01: Scaling foundation models with lightning attention, 2025. URL <https://arxiv.org/abs/2501.08313>.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.
- Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*, pp. 8732–8740. AAAI Press, 2020.
- Song, Y., Mi, Z., Xie, H., and Chen, H. Powerinfer: Fast large language model serving with a consumer-grade gpu, 2023.
- Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.
- Team, Q. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Wang, X., Zheng, Y., Wan, Z., and Zhang, M. Svdilm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.
- Xue, L., Fu, Y., Lu, Z., Mai, L., and Marina, M. Moe-infinity: Activation-aware expert offloading for efficient moe serving. *arXiv preprint arXiv:2401.14361*, 2024a.
- Xue, L., Fu, Y., Lu, Z., Mai, L., and Marina, M. Moe-infinity: Activation-aware expert offloading for efficient moe serving, 2024b.

Yang, C., Sui, Y., Xiao, J., Huang, L., Gong, Y., Duan, Y., Jia, W., Yin, M., Cheng, Y., and Yuan, B. Moe-i2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. *arXiv preprint arXiv:2411.01016*, 2024.

Yin, L., Wu, Y., Zhang, Z., Hsieh, C.-Y., Wang, Y., Jia, Y., Pechenizkiy, M., Liang, Y., Wang, Z., and Liu, S. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

Yuan, Z., Shang, Y., Song, Y., Wu, Q., Yan, Y., and Sun, G. ASVD: activation-aware singular value decomposition for compressing large language models. *CoRR*, abs/2312.05821, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A. M., Le, Q. V., Laudon, J., et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Appendix

Our appendix provides additional information and in-depth analysis to supplement the main content of the paper on MoE-SVD. It is organized into two main sections: further discussions and implementation details. The discussion section covers innovation, advantages and implications, disadvantages, and social implications of our proposed method. The implementation details section includes an algorithm table and specific implementation considerations.

A. Further Discussions

A.1. Ethics Statement

We focus solely on developing efficient techniques for Large Language Models (LLMs), utilizing publicly available datasets and models. Our research is not designed to address human ethics or privacy concerns directly. Instead, we concentrate on improving the computational efficiency and deployment capabilities of existing MoE LLMs, which may indirectly contribute to broader accessibility and utilization of these powerful models.

A.2. Reproducibility

We affirm the solid reproducibility of our results and provide specific code implementations in the appendix. Our main experiments represent average outcomes from multiple repetitions, ensuring reliability. MoE LLMs, being very large models, exhibit relatively small variances in experimental results and evaluations. To further demonstrate the robustness and repeatability of our method, we present detailed results for different initial seeds, showcasing consistent performance across various conditions.

A.3. Detailed Comparison with Other Work

As shown in Table 1, our MoE-SVD method focuses on large-scale MoE architectures and avoids fine-tuning while employing advanced strategies like activation-weighted SVD and selective decomposition. Other methods either target dense model (Dong et al., 2024; Li et al., 2024c;h), smaller architectures (Li et al., 2024d;e), require retraining, or address different aspects of model compression (*e.g.*, quantization (Dong et al., 2025b; 2023b), AutoML (Dong et al., 2025a; 2023a), distillation (Li et al., 2024a;b; 2023a; Li & Jin, 2022; Li, 2022)). In addition, we conducted comparisons with these methods where feasible and included the results in the revised manuscript. However, our primary comparisons are with SVD-based methods (ASVD, and SVD-LLM), which are directly applicable to MoE architectures and closely aligned with our methodology. These comparisons demonstrate the clear advantages of our approach. Finally, we extra individually discuss and contrast other methods primarily target different architectures, employ distinct strategies, or address challenges unrelated to those in our study:

- **STUN** (Lee et al., 2024) lacks available code or implementation details, as it was recently published on arXiv. Without the ability to replicate its results, it remains inaccessible for direct comparison.
- **Wanda** and **SparseGPT** are sparsity-based pruning methods designed for dense architectures, not Mixture-of-Experts (MoE) models. These methods do not utilize Singular Value Decomposition (SVD) or address the unique challenges of large-scale MoE architectures, such as expert diversity and weight redundancy.
- **LoSparse** and **MC-MoE** (Huang et al., 2024) focus on small-scale models, such as BERT-base (110M parameters) and Switch-base-32 (2.0B parameters), whereas our method is designed for large-scale MoE models (16B–57B parameters). Additionally, these methods rely on fine-tuning and knowledge distillation, whereas our approach avoids such computationally expensive steps.
- **MoE-Compression** (He et al., 2024) employs sparsity and layer-dropping techniques, which are fundamentally different from our SVD-based strategy. This framework focuses on compressing small to medium-sized models, making it less relevant to our focus on large-scale MoE architectures.

A.4. Detailed Comparison with Fine-Tuning Compression Methods

We also compare our method with fine-tuning-based approaches like **MC-SMoE**. Table 9 highlights the key differences between our method and MC-SMoE.

Table 9: Comparison of MoE-SVD with MC-SMoE (Li et al., 2024g).

Method	Comparison Method	Pipeline	Decomposition	Training	Advanced Strategy for MoE LLMs	Evaluated Models
MC-SMoE (Li et al., 2024g)	Model merge methods	Merge, then compress	Low-rank + sparse matrix	Needs fine-tuning	Frequency-based merge	TS-base (220M), Switch-base-32 (2.0B)
MoE-SVD (Ours)	Model SVD methods	Decompose, then trim	Low-rank matrix	No fine-tuning required	Activation-weighted SVD, selective decomposition, matrix sharing	Mixtral-8x7B/22B, Phi-3.5-MoE (41.9B), DeepSeekMoE (16.4B)

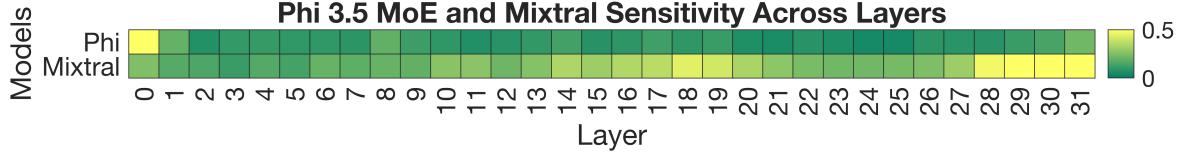


Figure 8: Sensitivity scores of Mixtral-8x7B and Phi-3.5-MoE accrossing layers

Our approach differs significantly from MC-SMoE in terms of methodology and application. While MC-SMoE relies on merging experts and fine-tuning, MoE-SVD employs selective decomposition tailored for large-scale MoE architectures without requiring fine-tuning. As shown in Table 9, our method achieves superior performance and scalability, handling models with up to 57B parameters.

A.5. Selective Decomposition Results

We detail the results of our selective decomposition strategy in Table 10. These results show clear trends: as the compression ratio increases, more layers are selectively decomposed. Middle layers are more likely to be decomposed, while the first and last layers tend to be retained, aligning with empirical observations and sensitivity analyses.

Table 10: Decomposed layers of our selective decomposition strategy by varying compression ratios.

Ratios	Mixtral-8x7B Selected Decomposition Expert Layers	Ratios	Phi3.5 MoE Selected Decomposition Expert Layers
20	[3,5,6,7,9,12,23,24,25],	20	[11,12,15,20,21,23,24,25],
30	[3,5,6,7,9,12,13,22,23,24,25,26],	30	[11,12,15,20,21,23,24,25,27,28],
40	[3,5,6,7,9,10,12,13,21,22,23,24,25,26],	40	[10,11,12,15,16,18,20,21,23,24,25,26,27,28],
50	[3,5,6,7,9,10,12,13,14,15,16,20,21,22,23,24,25,26],	50	[5,6,10,11,12,13,15,16,18,20,21,23,24,25,26,27,28],
60	[2,3,5,6,7,9,10,12,13,14,15,16,17,20,21,22,23,24,25,26,27]	60	[5,6,10,11,12,13,15,16,18,19,20,21,23,24,25,26,27,28,29]

B. More results

B.1. More Results on Per-layer Decomposition and Compression Ratios

Our extended experimental investigations provide deeper insights into the efficacy and behavior of the MoE-SVD compression technique. Figure 8 presents a detailed distribution of sensitivity scores across layers for both Mixtral-8x7B and Phi-3.5-MoE models. This analysis elucidates the varying impact of compression on different layers within the network architecture. Furthermore, we conduct an in-depth examination of perplexity results for each decomposed block of Mixtral-8x7B at 20% compression, as illustrated in Figure 7. These results offer valuable insights into the relationship between compression ratios and model performance. To further optimize our layer selection process, we develop and implement a sophisticated heatmap-based approach, visualized in Figure 10. This method provides a more intuitive and data-driven way to identify layers most suitable for compression, enhancing the overall efficiency and effectiveness of our MoE-SVD technique.

B.2. More Comparison with Structured Compression Methods

In Table 2, we compare our MoE-SVD against several structured compression methods applied to Mixtral-8x7B, as well as methods specifically targeting expert layer compression (Li et al., 2024g; He et al., 2024). Our MoE-SVD achieves a

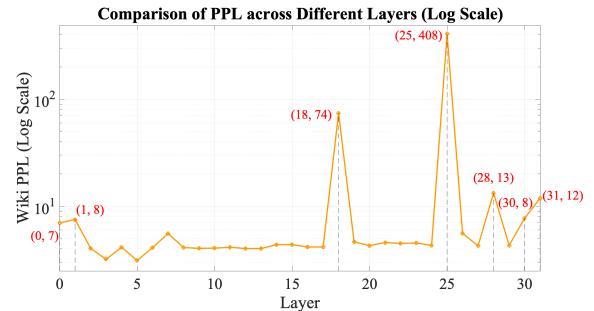


Figure 7: Perplexity of Mixtral-8x7B via 20% per-layer SVD decomposition on WikiText-2.

Table 11: Metrics (model size, TFLOPs, runtime) of MoE-SVD. Runtime denotes runtime throughput (Tokens/sec) on a single H800 GPU.

Mixtral-8x7B	Dense	20%	30%	40%	50%	60%
Model-size	46.7B	37.1B	32.2B	28.3B	23B	17.6B
TFLOPs	5.27E+14	4.92E+14	4.40E+14	4.26E+14	3.97E+14	3.67E+14
Runtime	87.73	104.66	106.03	108.83	123.88	156.1
Phi-3.5-MoE	Dense	20%	30%	40%	50%	60%
Model-size	41.9B	33.2B	29B	25.2B	20.6B	16.4B
TFLOPs	2.72E+14	2.46E+14	2.27E+14	2.00E+14	1.82E+14	1.71E+14
Runtime	98.2	108.63	114.8	124.79	137.17	148.7
DeepSeekMoE	Dense	20%	30%	40%	50%	60%
Model-size	6.4B	13.2B	11.4B	9.7B	8B	6.4B
TFLOPs	1.10E+14	1.02E+14	9.88E+13	9.29E+13	9.25E+13	8.82E+13
Runtime	52.53	62.79	94.71	118.93	119.81	128.71

Table 12: Significance test for three repeated experiments of Mixtral-8x7B and Phi-3.5-MoE compressed by MoE-SVD under 20% compression ratios.

Mixtral-8x7B	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA
MoE-SVD (512)	0.32±0.0199	0.78±0.0088	0.73±0.0129	0.57±0.0050	0.48±0.0146	0.79±0.0096	0.37±0.0087
MoE-SVD (512) +LoRA	0.33±0.0205	0.80±0.0086	0.73±0.0128	0.61±0.0049	0.55±0.0145	0.81±0.0095	0.38±0.0084
Phi-3.5-MoE	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA
MoE-SVD (512)	0.38±0.0215	0.76±0.0090	0.72±0.0127	0.63±0.0049	0.53±0.0146	0.77±0.0099	0.35±0.0081
MoE-SVD (512) +LoRA	0.39±0.0213	0.81±0.0080	0.74±0.0123	0.65±0.0041	0.54±0.0142	0.79±0.0095	0.36±0.0084

runtime speedup of 1.2× while maintaining performance across various benchmarks. Specifically, MoE-SVD records lower perplexities on language modeling tasks such as WikiText-2 (4.44) and PTB (15.21) compared to Wanda (4.72 and 18.8) and SparseGPT (4.61 and 21.11). On downstream tasks, our method attains the highest average score of 0.58, outperforming Unified-MoE-Compress (He et al., 2024)’s 0.54 and significantly surpassing LoSparse (Li et al., 2023c) and MC-SMoE (Li et al., 2024g), which exhibit substantial performance drops. These results highlight that MoE-SVD not only accelerates inference but also preserves or improves accuracy relative to other methods.

B.3. More Results on Real-time and Significance test

In Tables 11 and 12, we present the experimental results of our MoE-SVD method applied to Mixtral-8x7B and Phi-3.5-MoE models at 20% ratios. Our approach achieves substantial reductions in model size and computational overhead while maintaining competitive performance. Specifically, MoE-SVD reduces the model size of Mixtral-8x7B from 46.7B to 37.1B and improves runtime throughput from 87.73 to 104.66 Tokens/Sec. The significance tests in Table 12 indicate that these improvements are statistically meaningful across multiple runs.

B.4. Detailed Tensor Parallelism and Potential Communication Overhead

Our MoE-SVD method significantly reduces computation time for weight matrices through low-rank decomposition, achieving a 1.2x–1.5x speedup during inference. These results outperform existing structured compression methods on an algorithmic level, confirming the effectiveness of our approach. When combined with hardware optimization techniques such as tensor parallelism, all low-rank decomposition methods, including ours, MC-SMoE [1], ASVD, and SVD-LLM, inherently introduce communication overhead. This trade-off is a common phenomenon for such techniques in tensor-parallel scenarios. However, the total optimization between computation time and communication time depends on specific hardware configurations.

Our SVD-MoE framework optimizes the trade-off between computation and communication by leveraging low-rank matrix decomposition. The expert weight matrix $\mathbf{W} \in \mathbb{R}^{H \times H}$, where $H = 4096$, is decomposed into $\mathbf{U} \in \mathbb{R}^{H \times r}$ and $\mathbf{V}^\top \in \mathbb{R}^{r \times H}$, with the reduced rank $r = 1024$ representing 25% of the hidden dimension ($r = H/4$). For input activations

$\mathbf{X} \in \mathbb{R}^{B \times H}$ with batch size $B = 128$, the matrix multiplication $\mathbf{Y} = \mathbf{X}\mathbf{W}$ is replaced by a two-stage process, $\mathbf{Y} = \mathbf{X}\mathbf{U}\mathbf{V}^\top$, introducing an intermediate activation $\mathbf{Z} = \mathbf{X}\mathbf{U} \in \mathbb{R}^{B \times r}$.

The computational efficiency is evident in the following metrics. The original computation requires:

$$\text{Computations}_{\text{before}} = B \times H^2 = 128 \times 4096^2 = 2.15 \times 10^{11} \text{ operations.}$$

After decomposition, the computation is reduced to:

$$\text{Computations}_{\text{after}} = 2 \times B \times H \times r = 2 \times 128 \times 4096 \times 1024 = 1.07 \times 10^{11} \text{ operations.}$$

This results in a 50% reduction in computational cost. However, the decomposition increases communication volume from $B \times H = 128 \times 4096 = 524,288$ elements to $B \times (r + H) = 128 \times (1024 + 4096) = 655,360$ elements, representing a 25% increase.

To analyze this trade-off, we evaluate performance on NVIDIA A100 GPUs, which feature 600 GB/s NVLink bandwidth and a 19.5 TFLOPS peak performance (FP32). Communication time is calculated as:

$$\text{Communication Time} = \frac{\text{Total Data (bytes)}}{\text{Bandwidth (bytes/s)}}.$$

The communication time increases from 3.5×10^{-6} seconds to 4.37×10^{-6} seconds. Meanwhile, computation time decreases from:

$$\frac{\text{Computations}_{\text{before}}}{19.5 \times 10^{12}} \approx 11 \times 10^{-3} \text{ seconds,}$$

to:

$$\frac{\text{Computations}_{\text{after}}}{19.5 \times 10^{12}} \approx 5.5 \times 10^{-3} \text{ seconds.}$$

The total processing time improves from 11.0035×10^{-3} seconds to 5.50437×10^{-3} seconds, yielding a net optimization of 5.49913×10^{-3} seconds.

The marginal increase in communication overhead:

$$\Delta \text{Communication Time} = 0.87 \times 10^{-6} \text{ seconds,}$$

is negligible compared to the substantial reduction in computation time:

$$\Delta \text{Computation Time} = 5.5 \times 10^{-3} \text{ seconds.}$$

This demonstrates that the SVD-MoE architecture effectively navigates the trade-off between computational efficiency and communication overhead. The high-bandwidth infrastructure ensures that increased communication does not negate the performance gains achieved through decomposition.

We considered conducting additional experiments to further validate these findings. However, performing such experiments was infeasible within the rebuttal period due to the limitations of PyTorch’s tensor parallelism, which only supports official models. Adapting tensor parallelism for non-official models like ours would require substantial modifications to the library. Nonetheless, our analysis strongly suggests that the computational benefits of SVD-MoE outweigh the communication costs. We plan to explore these aspects in future work to further optimize the performance of decomposed MoE architectures.

C. Pseudocode

In our experimental implementation, we present a detailed algorithmic procedure for compressing MoE-based large language models using the proposed MoE-SVD method. Algorithm 4 outlines the main steps of this approach. The process begins by collecting scaling matrices through forward hooks during inference, as shown in Algorithm 1 (Step 1). This step is crucial for capturing activation patterns and computing the sensitivity metric for each expert. Subsequently, we perform singular value decomposition (SVD) on the scaled weight matrices, followed by truncation for effective compression, as detailed in Algorithm 2 (Step 2). Our method introduces a V-matrix sharing mechanism, where the most frequently used V-matrix is selected and shared among all experts, as described in Algorithm 3 (Step 3). Additionally, we employ U-matrix trimming by retaining the top- k U-matrices based on expert sampling frequencies to refine the expert functions (Step 4). To ensure numerical stability, we apply the adjustment function provided in Algorithm 5, which modifies matrices to be positive definite when necessary. This comprehensive approach enables significant model compression while maintaining performance, effectively addressing the need for efficient large-scale language models.

D. Implementation Details

MoE-SVD optimizes MoE models by selectively decomposing less critical experts to reduce computational complexity while maintaining performance. It consists of two main phases: computing a sensitivity metric S_L for each expert layer during calibration data inference, and decomposing experts.

D.1. Calculation of Sensitivity Score

During calibration data inference, both the sensitivity metric $S_L = \sum_{i=1}^N f_i \cdot p_i \cdot a_i$ and the activation matrices for each expert i are collected. The sensitivity metric integrates utilization frequency f_i , principal Rank p_i , and activation outliers a_i , where each component is described in detail below:

Sampling Frequency (f_i): The variable f_i represents the utilization frequency of the i -th expert, quantifying how often this expert is selected by the router during inference. It is calculated over a calibration dataset \mathcal{X} as:

$$f_i = \frac{\sum_{x \in \mathcal{X}} \mathbb{I}[i \in \text{TopK}(G(x), k)]}{|\mathcal{X}|}, \quad (9)$$

where $G(x)$ is the output of the gating network for input x , $\text{TopK}(G(x), k)$ returns the indices of the top k selected experts, $\mathbb{I}[\cdot]$ is the indicator function, and $|\mathcal{X}|$ denotes the total number of samples in the dataset. This metric reflects the relative importance of each expert based on its selection frequency.

Principal Rank (p_i): The variable p_i denotes the principal rank of the i -th expert, which is the number of dominant singular values in the diagonal matrix Σ_i obtained from the SVD of the expert's weight matrix W_i . p_i is defined as the number of singular values in Σ_i that exceed a given threshold, effectively capturing the dimensionality of the weight matrix's significant components. This rank reflects the structural complexity of the expert's weight representation, with higher values of p_i indicating more complex and information-rich weights.

Activation Outliers (a_i): The variable a_i measures the proportion of activations in the i -th expert that exceed a certain threshold relative to the mean absolute activation value. For a set of activations A_i in the i -th expert, a_i is computed as:

$$a_i = \frac{\sum_{a \in A_i} \mathbb{I}(|a| > \tau \cdot \text{Mean}(|A_i|))}{|A_i|}, \quad (10)$$

where $|A_i|$ denotes the total number of activations for the i -th expert, $\text{Mean}(|A_i|)$ is the mean absolute value of these activations, and τ is a user-defined threshold. This metric highlights the presence of outlier activations indicative of the expert's contribution to the model's capacity. The overall sensitivity metric S_L aggregates these factors across all N experts in a layer, providing a comprehensive measure of the layer's importance.

D.2. Decomposition Process of Expert Matrix

Following ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2024), our framework employs an activation-weighted SVD that enhances the vanilla SVD by incorporating activation statistics to improve decomposition accuracy. With activation matrix X and original weight W_{original} , we compute the activation-weighted matrix by scaling the original weight matrix based on the activation statistics:

$$W_{\text{aw}} = W_{\text{original}} \cdot S, \quad (11)$$

where $W_{\text{aw}} \in \mathbb{R}^{m \times n}$ represents the activation-weighted matrix and matrix S is obtained through cholesky decomposition of activation gram matrix $\mathbf{X}\mathbf{X}^T$. We then perform SVD on W_{aw} and final compressed weight matrix is obtained by truncating the smallest singular values:

$$W_{\text{aw}} = U \cdot \text{Trunc}(\Sigma) \cdot V^T \cdot S^{-1}, \quad (12)$$

This activation-weighted approach effectively mitigates reconstruction loss from outliers during matrix decomposition while maintaining the essential characteristics of the original weight distribution.

D.3. Model-Specific Configurations

In our experimental realization, we develop tailored post-decomposition strategies for various large language models, each with unique architectures. For Mixtral-8x7B, which employs 8 experts per layer, we implement a novel approach of sharing

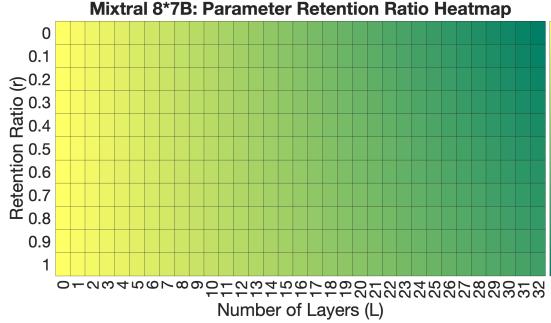


Figure 9: Retained parameters calculation for Mixtral-8×22B.

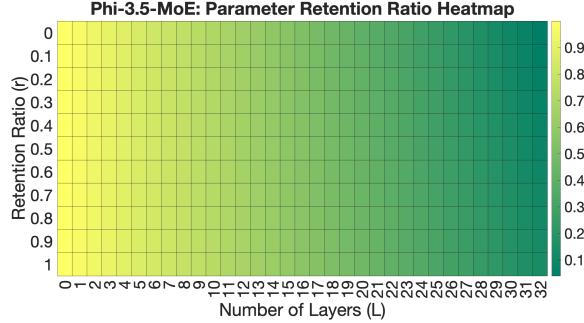


Figure 10: Retained parameters calculation for Phi-3.5-MoE.

V components (v_1, v_2, v_3) from the most frequently activated expert while retaining U components (u_1, u_2, u_3) from the top two most frequent experts. We extend this methodology to Phi-3.5-MoE, featuring 16 experts per layer, by broadening the retention scope. In this instance, we share V components from the most frequently selected expert and preserve U components from the top four most frequent experts. For the more complex Deepseek 16B, which utilizes 64 experts per block, we innovate further by partitioning the experts into 8 distinct groups. Within each group, we share the most frequent V component and strategically trim the 6 lowest frequency U components. This carefully crafted selective retention and sharing approach enables us to maintain model performance while achieving substantial reductions in both parameter count and computational requirements.

D.4. Computational Efficiency

We rigorously assess the computational efficiency of our LLM compression techniques, recognizing its paramount importance for practical deployment scenarios. Our MoE-SVD compression methodology comprises two distinct phases: activation data collection and SVD decomposition with expert trimming. Through extensive experimentation on the Phi 3.5 MoE model, we meticulously quantify time requirements for various layer counts. Our findings reveal that single-layer processing consumes 266 seconds for activation collection and 107 seconds for SVD and trimming. These durations exhibit a non-linear increase, reaching 489 and 209 seconds for two layers, and 689 and 320 seconds for three layers, respectively. In a comprehensive 20% layer compression test, we observe a total time requirement of 44 minutes, with 30 minutes allocated to data collection and 14 minutes to SVD and trimming. These results provide crucial insights into the scalability and efficiency of our approach across different model configurations.

D.5. Scalability Analysis

Our in-depth scalability analysis unveils intriguing patterns in the computational behavior of our compression technique. The activation collection phase demonstrates near-linear time growth with respect to layer count, indicating limited parallelization potential due to the inherent sequential nature of data propagation. However, we emphasize that this collection process is a one-time operation per model, with the resulting data being storable and reusable for various compression ratios. In contrast, the SVD and trimming phase exhibits promising sub-linear scaling, suggesting enhanced opportunities for parallelization. While our current implementation relies on sequential Python loops, we identify significant potential for efficiency improvements through parallel processing of experts across layers. This aligns seamlessly with the independent operation of experts in different layers of MoE models, indicating promising scalability prospects for MoE-SVD, particularly in the computationally intensive SVD and trimming phase when applied to large-scale models.

D.6. Potential Parallelization Strategies

To further optimize our approach, we explore a range of potential parallelization strategies. We consider leveraging Python’s multiprocessing modules and GPU acceleration frameworks such as PyTorch or TensorFlow to exploit parallel computing capabilities. For models exceeding 100B parameters, we propose the utilization of distributed computing frameworks like Dask or Ray to efficiently scale computation across multiple machines. We hypothesize that this approach could potentially reduce SVD phase time complexity from $O(mn^2)$ to near-linear relative to processor count, with the potential to scale with

the maximum expert count per layer rather than the total layer count. However, we acknowledge that the effectiveness of these strategies may vary based on available computational resources, inter-process communication overhead, and the challenges of expert load balancing in distributed environments. In addressing the research challenges associated with our proposed parallelization strategies, we encounter several non-trivial technical hurdles. These include the need to fundamentally redesign algorithms for efficient concurrent processing, develop robust mechanisms for managing complex data dependencies, and optimize resource utilization across heterogeneous computing environments. We emphasize the critical importance of conducting comprehensive empirical studies to quantify potential performance improvements across a diverse range of model sizes and hardware configurations. While we anticipate that parallel processing may significantly enhance MoE-SVD's scalability for large language models, we maintain a cautious stance regarding its effectiveness when combined with our existing matrix sharing and trimming optimizations. We assert that rigorous experimentation and thorough analysis are essential to verify these potential benefits and to fully understand the implications of our proposed parallelization strategies in real-world, large-scale language model compression scenarios.

Algorithm 1 PyTorch code for sensitivity metric of MoE-SVD.

```

import torch
import torch.nn as nn

def compute_layer_sensitivity(experts_weights, activations, gating_outputs, calibration_data, top_k=2,
    tau=2.0):
    """
    Compute layer-wise sensitivity metric S_L for MoE compression

    Args:
        experts_weights (list of torch.Tensor): Weight matrices for each expert
        activations (list of torch.Tensor): Activation values for each expert
        gating_outputs (torch.Tensor): Router outputs for calibration data
        calibration_data (torch.Tensor): Calibration dataset
        top_k (int): Number of experts to select per token
        tau (float): Threshold for activation outliers

    Returns:
        float: Layer sensitivity score S_L
    """
    num_experts = len(experts_weights)
    device = experts_weights[0].device

    # Compute sampling frequency (f_i)
    top_k_indices = torch.topk(gating_outputs, top_k, dim=-1).indices
    expert_counts = torch.zeros(num_experts, device=device)
    for indices in top_k_indices:
        expert_counts[indices] += 1
    f_i = expert_counts / len(calibration_data)

    # Compute principal rank (r_i) using SVD
    r_i = torch.zeros(num_experts, device=device)
    for i, weight in enumerate(experts_weights):
        U, S, V = torch.linalg.svd(weight)
        # Count singular values above threshold
        threshold = torch.max(S) * 1e-2 # Example threshold
        r_i[i] = torch.sum(S > threshold)

    # Compute activation outliers (a_i)
    a_i = torch.zeros(num_experts, device=device)
    for i, activation in enumerate(activations):
        mean_abs_act = torch.mean(torch.abs(activation))
        outliers = torch.sum(torch.abs(activation) > tau * mean_abs_act)
        a_i[i] = outliers / activation.numel()

    # Compute final sensitivity metric S_L
    S_L = torch.sum(f_i * r_i * a_i)

    return S_L

```

Algorithm 2 PyTorch code for SVD Expert Decomposition of MoE-SVD.

```

class MoESVDCompression:
    def __init__(self, truncate_k=None, top_k_experts=2):
        """
        Initialize Activation-Weighted SVD with Matrix Sharing and Trimming

        Args:
            truncate_k (int): Number of singular values to keep
            top_k_experts (int): Number of top experts to select for U-matrix trimming
        """
        self.truncate_k = truncate_k
        self.top_k_experts = top_k_experts

    def compute_activation_weights(self, X):
        """
        Compute activation-weighted scaling matrix S using Cholesky decomposition

        Args:
            X (torch.Tensor): Activation matrix [batch_size, feature_dim]
            torch.mm(X, X.t()) is Cumulative activation matrix, representing the sum of processed
            activation data.
        """
        # Compute Gram matrix
        gram = torch.mm(X, X.t())

        # Cholesky decomposition
        S = torch.linalg.cholesky(gram)
        return S

    def decompose_expert(self, W_original, X):
        """
        Perform activation-weighted SVD on single expert

        Args:
            W_original (torch.Tensor): Original weight matrix
            X (torch.Tensor): Activation matrix
        """
        # Compute activation-weighted matrix
        S = self.compute_activation_weights(X)
        W_aw = torch.mm(W_original, S)

        # Perform SVD
        U, sigma, V = torch.linalg.svd(W_aw, full_matrices=False)

        # Truncate if specified
        if self.truncate_k is not None:
            U = U[:, :self.truncate_k]
            sigma = sigma[:self.truncate_k]
            V = V[:self.truncate_k, :]

        return U, torch.diag(sigma), V, S

```

Algorithm 3 PyTorch code for Matrix Sharing & Trimming of MoE-SVD

```

class MoESVDCompression:
    def __init__(self, truncate_k=None, top_k_experts=2):
        """
        Initialize Activation-Weighted SVD with Matrix Sharing and Trimming

        Args:
            truncate_k (int): Number of singular values to keep
            top_k_experts (int): Number of top experts to select for U-matrix trimming
        """
        self.truncate_k = truncate_k
        self.top_k_experts = top_k_experts
    def __init__(self, truncate_k=None, top_k_experts=2):
        """
        Initialize Activation-Weighted SVD with Matrix Sharing and Trimming

        Args:
            truncate_k (int): Number of singular values to keep
            top_k_experts (int): Number of top experts to select for U-matrix trimming
        """
        self.truncate_k = truncate_k
        self.top_k_experts = top_k_experts

    def compress_moe(self, expert_weights, activations, routing_frequencies):
        """
        Compress MoE using V-matrix sharing and U-matrix trimming

        Args:
            expert_weights (list): List of expert weight matrices
            activations (list): List of activation matrices for each expert
            routing_frequencies (torch.Tensor): Expert selection frequencies
        """
        num_experts = len(expert_weights)
        compressedExperts = []

        # Decompose all experts
        decomposed = []
        for i in range(num_experts):
            U, Sigma, V, S = self.decompose_expert(expert_weights[i], activations[i])
            decomposed.append((U, Sigma, V, S))

        # Select shared V-matrix based on highest routing frequency
        max_freq_idx = torch.argmax(routing_frequencies)
        V_shared = decomposed[max_freq_idx][2]

        # Sort experts by routing frequency for U-matrix trimming
        sorted_indices = torch.argsort(routing_frequencies, descending=True)

        # Perform U-matrix trimming and construct compressed experts
        for i in range(num_experts):
            # Find top-k U-matrices from more frequently used experts
            more_frequent = [j for j in sorted_indices if routing_frequencies[j] > routing_frequencies[i]
            ]
            top_k_indices = more_frequent[:self.top_k_experts]

            if len(top_k_indices) < self.top_k_experts:
                # If not enough more frequent experts, use own U-matrix
                top_k_indices = top_k_indices + [i]

            # Combine selected U-matrices and corresponding Sigma matrices
            U_combined = torch.zeros_like(decomposed[i][0])
            Sigma_combined = torch.zeros_like(decomposed[i][1])

            for idx, expert_idx in enumerate(top_k_indices[:self.top_k_experts]):
                U_combined += decomposed[expert_idx][0]
                Sigma_combined += decomposed[expert_idx][1]

            # Reconstruct compressed expert
            W_compressed = torch.mm(torch.mm(U_combined, Sigma_combined),
                                   torch.mm(V_shared, torch.inverse(decomposed[i][3])))
            compressedExperts.append(W_compressed)

    return compressedExperts

```

Algorithm 4 SVD-MOE: Expert Decomposition, V-Matrix Sharing, and U-Matrix Trimming

Input: model: Language model, data: Calibration data, device: Device, layers: Selected layers, config: Expert configuration
Output: model_compressed: Compressed model

{Step 1: Calibration Processing}

```

for layer  $\ell$  in layers do
    for expert  $i$  in  $\ell$  do
         $f_i \leftarrow \frac{i \in \text{TopK}(G(x), K)}{|\mathcal{X}|};$ 
        {Update activation matrix}
         $S_i \leftarrow S_i + AA^T;$ 
         $U, \Sigma, V^T \leftarrow \text{SVD}(\text{expert}_i);$ 
         $r_i \leftarrow \text{Rank}(\Sigma);$ 
        {Update weight outlier}
         $\bar{W}_i \leftarrow \text{mean}(\text{expert}_i.W);$ 
         $a_i \leftarrow +\frac{\#\{|A| > \alpha \bar{W}_i\}}{A};$ 
         $\text{expert}_i.S_L \leftarrow \text{expert}_i.S_L + f_i r_i a_i;$ 
    
```

{Step 2: Decompose Experts}

```

for layer  $\ell$  in layers do
    for expert  $i$  in  $\ell$  do
        if  $\text{expert}_i.S_L < \tau$  then
            if  $S_i$  is not positive definite then
                Adjust  $S_i$ 
             $W_s \leftarrow \text{expert}_i.W \times S_i;$ 
             $U, \Sigma, V^T \leftarrow \text{SVD}(W_s);$ 
             $\Sigma_{trunc} \leftarrow \text{Truncate}(\Sigma);$ 
             $\text{expert}_i.U \leftarrow U\Sigma_{trunc};$ 
             $\text{expert}_i.V \leftarrow V^T;$ 
    
```

{Step 3: V-Matrix Sharing}

{Select shared V-matrix}

$V_s \leftarrow \arg \max_{V_i} f(V_i);$

{Select top-2 U-matrices}

$\{U_{i,1}, U_{i,2}\} \leftarrow \text{TopK}(\{U_j\}, k=2);$

{Step 4: U-Matrix Trimming and Selection}

{Update expert function}

$E_i(x) \leftarrow (U_{i,1}\Sigma_{i,1} + U_{i,2}\Sigma_{i,2})V_s^T x;$

Algorithm 5 MakePositiveDefinite: Adjust Matrix to be Positive Definite

Function MakePositiveDefinite (M , tolerance, max_attempts) :

Input: M - Input matrix; tolerance - Small value for adjustment; max_attempts - Maximum number of attempts

Output: M_{pd} - Positive definite matrix

Step 1: Symmetrize the Matrix $M_{sym} \leftarrow \frac{M+M^T}{2}$ // Ensure the matrix is symmetric

Step 2: Check Eigenvalues Compute eigenvalues λ of M_{sym}

if any $\lambda_i < 0$ **then**

$\lambda \leftarrow \lambda + |\min(\lambda_i)| + \text{tolerance}$ // Shift negative eigenvalues to positive

end

Step 3: Reconstruct Positive Definite Matrix $M_{pd} = V \text{diag}(\lambda) V^T$ where V are the eigenvectors of M_{sym}

Step 4: Ensure Matrix is Symmetric $M_{pd} \leftarrow \frac{M_{pd} + M_{pd}^T}{2}$

return M_{pd}