TD-MoE: Tensor Decomposition for MoE Models

Anonymous authors
Paper under double-blind review

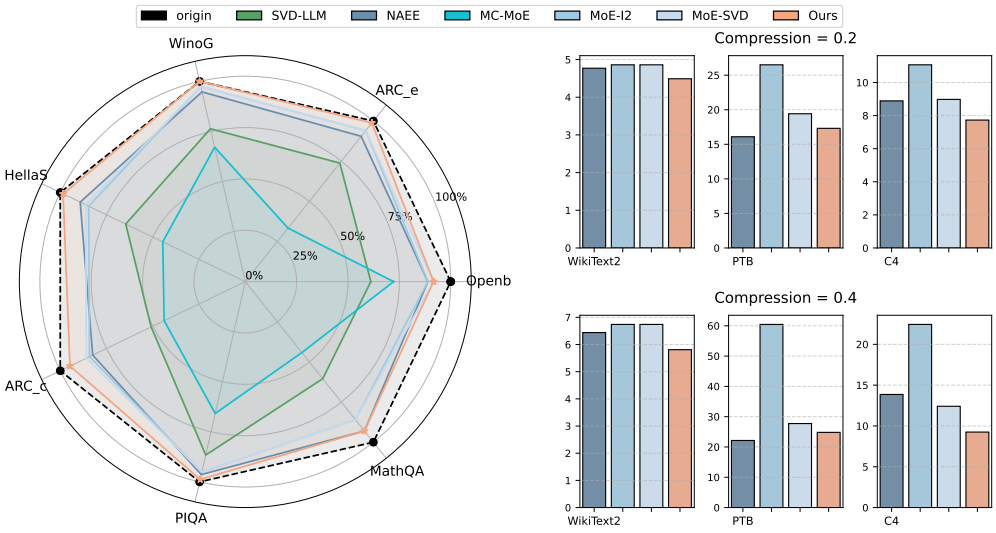
Abstract

Mixture-of-Experts (MoE) architectures have demonstrated remarkable capabilities and scalability for large language models, but incur a prohibitive memory footprint due to duplicated expert parameters. Existing compression approaches, particularly those based on low-rank decomposition, typically operate at the granularity of individual experts. However, such per-expert methods overlook structural redundancies across experts, limiting their compression efficiency and effectiveness. In this work, we introduce TD-MoE (Tensor Decomposition for MoE Compression), a data-aware framework that jointly and holistically factorizes expert weights. Our contributions are threefold: (i) Cross-expert tensorization with joint 3D decomposition, which unifies all experts within a layer into a single tensor and captures shared structure beyond per-expert scope; (ii) A multi-linear whitening strategy, which decorrelates input and output features, yielding a more balanced and data-adaptive decomposition; (iii) A 3D rank allocation mechanism, which dynamically assigns 3D decomposition ranks across dimensions to best meet a target compression ratio while minimizing the reconstruction error. Extensive experiments on Qwen2-57B-A14 and Mixtral-8×7B across seven commonsense reasoning benchmarks demonstrate that TD-MoE achieves almost lossless performance under 20% parameter reduction, and delivers more than 11% and 14% gains over state-of-the-art decomposition-based baselines at 40% and 60% compression. Further ablation studies validate the effectiveness of each component, highlighting the importance of joint factorization, whitening, and rank allocation. The code is available at https://anonymous.4open.science/r/TD-MoE.

1 Introduction

Mixture-of-Experts (MoE) architectures are a key technique for scaling large language models to trillions of parameters while maintaining computational efficiency (Cai et al., 2025). By routing tokens to only a subset of experts, they achieve strong scalability, but at the cost of substantial memory overhead from duplicated expert parameters, which severely limits deployment efficiency in practical large-scale MoE systems.

A natural way to mitigate this overhead is through model compression, and recent research has explored a broad spectrum of strategies, including pruning (Lu et al., 2024), decomposition (Yuan et al., 2023; Wang et al., 2024; Li et al., 2025), merging (Zhou et al., 2025), and quantization (Hu et al., 2025). Pruning-based approaches reduce model size by removing redundant experts or entire MoE layers, while merging and subspace representations fuse functionally similar experts to cut parameter counts. Quantization methods (Huang et al., 2025) further reduce memory usage by lowering weight precision through adaptive bit-width allocation. Among these, decomposition-based techniques have emerged as a compelling direction. By factorizing the large weight matrices inside experts into low-rank components, decomposition can directly target the densest parameter blocks, yielding substantial compression without altering the model architecture or routing behavior. This property makes decomposition highly scalable and especially appealing for trillion-parameter MoE models, where the majority of memory cost arises from expert weights. A representative example is MoE-SVD (Li et al., 2025), which applies singular value decomposition (SVD) to each



(a) Accuracy across reasoning and understanding tasks.

(b) Perplexity(↓) breakdown per dataset.

Figure 1: **Comprehensive evaluation on compressing Mixtral-8×7B**. The left figure (a) shows accuracy across reasoning and understanding benchmarks at 20% compression, while the right figure (b) reports perplexity on Wikitext-2, PTB, and C4 under 20% and 40% compression.

expert's weight matrices, effectively reducing the number of parameters while maintaining model accuracy. Empirical studies show that MoE-SVD achieves notable compression ratios with limited perplexity degradation, underscoring the promise of decomposition for large-scale MoE compression. However, most existing decomposition approaches still operate at the level of individual experts Yang et al. (2024); Li et al. (2025), overlooking the structural redundancies across experts. This per-expert isolation limits their ability to fully leverage shared structure, resulting in suboptimal trade-offs between compression and model performance.

Building on these observations, we advocate that exploiting cross-expert structural redundancy is crucial for achieving more effective compression in MoE models. Experts within the same layer are often trained on related distributions and exhibit correlated weight patterns; yet, per-expert decomposition strategies, such as MoE-SVD, treat them independently. This motivates a shift from isolated expert-level factorization toward a joint tensor decomposition perspective that can capture correlations across experts in a unified manner.

To this end, we propose TD-MoE (Tensor Decomposition for MoE Compression), a data-aware framework that reformulates MoE compression as a joint tensor decomposition problem. Specifically, we stack the weights of all experts in a layer into a 3-dimensional tensor and apply Tucker decomposition to compress them jointly. To further enhance the decomposition quality, we introduce a multi-linear whitening strategy that decorrelates input and output dimensions before factorization, enabling more balanced and data-adaptive low-rank approximations. In addition, we design a 3D rank allocation mechanism that automatically distributes compression ranks across dimensions, allowing the method to meet a target compression budget. Extensive experiments demonstrate the effectiveness of TD-MoE. On Qwen2-57B-A14B and Mixtral-8×7B, TD-MoE achieves nearly lossless under 20% compression. At more aggressive settings of 40% and 60% compression, TD-MoE delivers over 11% and 14% relative improvement compared to MoE-SVD (Li et al., 2025) and NAEE Lu et al. (2024) across multiple commonsense reasoning and language modeling tasks. Ablation studies further validate the necessity of each component, highlighting the central role of cross-expert factorization, whitening, and rank allocation. Our contributions are summarized as follows:

- We propose TD-MoE, reformulating MoE compression from per-expert decomposition to a joint 3D tensor decomposition. By stacking all experts in a layer into a three-dimensional tensor, TD-MoE performs cross-expert compression that leverages shared structures and correlations overlooked by expert-wise methods.
- We propose a novel multi-linear whitening strategy that decorrelates features across dimensions, thereby balancing the contribution of different modes and enabling joint decomposition to identify more informative low-rank structures, ultimately achieving better trade-offs between compression budget and accuracy.
- Experiments demonstrate that TD-MoE delivers superior performance in compressing MoE models. In particular, it delivers nearly lossless accuracy at a 20% compression ratio, and outperforms state-of-the-art baselines such as MoE-SVD and NAEE by more than 14% under 40–60% compression.

2 Related Work

Recent advances have driven progress in compressing MoE models, aiming to enhance both efficiency and practical deployability. In the following, we review existing approaches and discuss their limitations.

2.1 Model Compression for Mixture-of-Experts

The large memory footprint of large-scale MoE models has recently brought the research into specialized compression techniques. These emerging methods primarily fall into three categories: expert pruning, expert merging, and low-rank decomposition, often used in combination. A primary strategy to reduce the MoE parameter count is to decrease the number of experts. This can be achieved by either removing experts entirely via expert pruning or combining similar ones via expert merging. The MC-SMoE framework (Li et al., 2024) exemplifies the merging approach by first clustering experts to identify similarity, then applying a frequency-based merging criterion to combine them. MoE-I2 (Yang et al., 2024) adopts a two-stage hybrid strategy, beginning with inter-expert pruning to discard less critical experts wholesale, followed by compressing the remaining inter-experts by low-rank decomposition. D2-MoE (Gu et al., 2025) first extracts a shared expert, decomposes the computed delta weights via SVD decomposition, and last applies quantization. These methods effectively reduce the expert count based on heuristic criteria, such as similarity, activation frequency, or explicit knowledge sharing. MoE-SVD (Li et al., 2025) enhances per-expert SVD by sharing the input projection matrix across experts and trimming the output projection matrix. This method relies on the strong assumption that expert redundancy is primarily concentrated in a shared input projection space, while functional specialization occurs only in the mapping to the output space. All the above methods impose a complex decision-making process, which requires a decision whether to keep, merge, or discard experts. Inversely, our approach offers an alternative solution: instead of removing experts, the tensor decomposition naturally learns a compressed representation in the expert dimension via its expert-mode factor matrix, allowing for a more graceful and data-driven reduction of expert-level redundancy.

2.2 Tensor Decomposition

Tensor decomposition provides a principled framework for exploiting multi-linear structure in high-dimensional parameter spaces. Classical methods include CP decomposition (Hitchcock, 1927), Tucker decomposition (Tucker, 1966), and Tensor Train (TT) (Oseledets, 2011). These techniques factorize a high-order tensor into a set of factor matrices and, in the case of Tucker, a core tensor, thereby reducing storage while retaining expressive capacity. In deep learning, tensor decomposition has been extensively studied as a technique for model compression. For instance, Stable Low-rank Tensor Decomposition (Phan et al., 2020) improves robustness when compressing convolutional layers; BATUDE (Yin et al., 2022) introduces a budget-aware Tucker decomposition to trade off accuracy and efficiency under explicit resource constraints; and Tucker-based CNN compression methods (Kim et al., 2015) demonstrate that convolutional kernels can be effectively reshaped into higher-order tensors for factorization. Similarly, Novikov et al. (Novikov

et al., 2015) apply tensorization to recurrent networks. Authors (Zhen et al., 2022) compress object detection models using Tucker decomposition by decomposing FFNs into multiple low-rank matrices in a chain. Despite these advances, most applications of tensor decomposition operate on individual weight matrices by reshaping them into higher-order tensors before applying CP, Tucker, or TT decomposition. This design is well-suited for CNN kernels, which naturally possess a three-dimensional structure, including spatial height, spatial width, and channel dimension. However, it does not directly translate to transformer-based MoE models, where experts are independent weight matrices without an inherent higher-order organization. As a result, conventional decomposition methods overlook structural redundancy that arises across multiple related weight matrices, such as the ensemble of experts within an MoE layer. Our work departs from this conventional perspective by introducing a cross-expert tensorization that explicitly stacks expert weights into a three-dimensional tensor.

3 METHOD

We introduce Tensor Decomposition for MoE (TD-MoE), method that compresses MoE layers through joint tensor decomposition. It consists of three components: cross-expert tensorization, multi-linear whitening, and 3D rank allocation. The pipeline operates as follows: given the weight matrices of all experts in a layer, we first stack them into a three-dimensional tensor that captures both expert diversity and cross-expert correlations. Next, we apply a whitening transformation along the input or output modes using activation statistics, which decorrelates feature dimensions and makes the subsequent decomposition more balanced and data-adaptive. We then perform Tucker decomposition on the whitened tensor to obtain a compact core tensor and factor matrices, with their sizes determined by an adaptive 3D rank allocation scheme that respects a global compression budget. Finally, the decomposed factors are re-colored via the inverse whitening matrices to reconstruct the compressed expert weights. The main framework is illustrated in Figure 2.

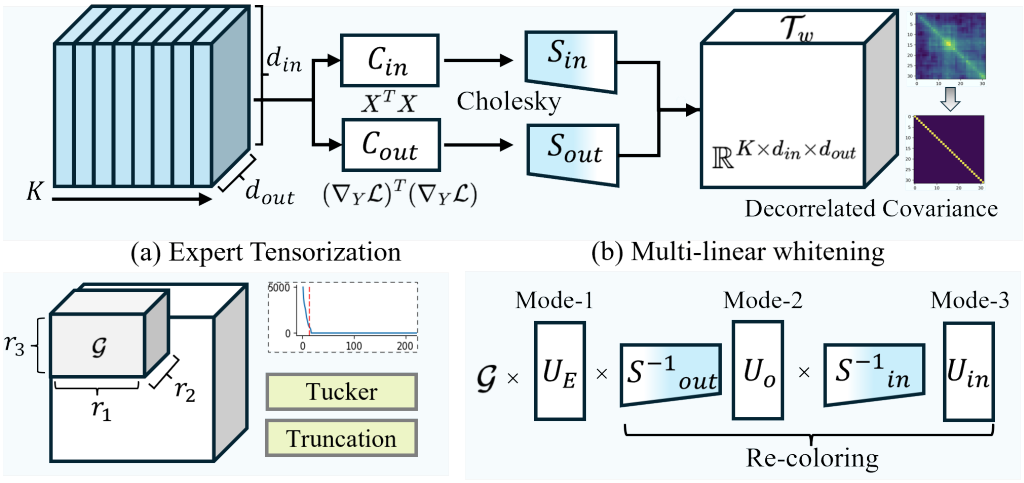


Figure 2: **Overall framework of TD-MoE.** (a) Expert Tensorization: expert weights are stacked into a 3D tensor $\mathcal{T}$; (b) Multi-linear Whitening: input/output modes are whitened to remove cross-dimensional correlations, producing $\mathcal{T}_w$; (c) Tucker Decomposition: $\mathcal{T}_w$ is factorized into a low-rank core $\mathcal{G}$ with ranks (r_1, r_2, r_3); (d) Reconstruction: factor matrices U_o and U_{in} are re-colored via inverse whitening to obtain the final low-rank approximation.

3.1 Problem Formulation

We formalize the compression objective for an MoE layer with K experts, where the i-th expert is parameterized by $W^{(i)} \in \mathbb{R}^{d_{in} \times d_{out}}$. Given a calibration distribution $\mathcal{D}_{\text{calib}}$, the goal is to find compressed weights $\{\hat{W}^{(i)}\}$ that preserve the activation behavior of the original model:

$$\min_{\{\hat{W}^{(i)}\}} \mathbb{E}_{x \sim \mathcal{D}_{\text{calib}}} \left[\sum_{i=1}^{K} \|W^{(i)} x - \hat{W}^{(i)} x\|_2^2 \right]. \tag{1}$$

Existing methods, such as MoE-SVD, apply low-rank decomposition independently to each $W^{(i)}$, truncating its factors to fit a parameter budget. This per-expert isolation overlooks redundancies across experts, thus limiting compression efficiency and leading to sharp performance drops at higher compression ratios.

3.2 Joint Tucker Factorization

Cross-Expert Tensorization. Instead of compressing experts individually, we adopt a global, ensemble-level view. We stack the K expert matrices into a three-dimensional tensor:

$$\mathcal{T} \in \mathbb{R}^{K \times d_{in} \times d_{out}},$$

where mode-1 indexes experts, mode-2 represents input features, and mode-3 represents output features. This tensorization unifies the ensemble into a single object, enabling joint modeling of intra-expert structure and inter-expert redundancies. Unlike pruning or merging, which make hard expert-level decisions, this formulation provides a principled basis for multi-linear factorization and data-aware compression.

Tucker Decomposition. Tucker decomposition generalizes principal component analysis to higher-order tensors. Given $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, it factorizes the tensor into a compact core $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{d_n \times r_n}$ for $n = 1, 2, 3$:

$$\mathcal{T}_w \approx \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \tag{2}$$

where the ranks (r_1, r_2, r_3) control both compression and reconstruction fidelity[1]. For MoE compression, we jointly model all experts as a three-dimensional tensor $\mathcal{T} \in \mathbb{R}^{K \times d_{in} \times d_{out}}$, where K is the number of experts. Instead of decomposing each expert in isolation, we apply Tucker decomposition directly to the whitened tensor $\mathcal{T}_w$, yielding:

$$\min_{\mathcal{G}, U_1, U_2, U_3} \|\mathcal{T}_w - \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3\|_F^2, \tag{3}$$

where $U_1 \in \mathbb{R}^{K \times r_1}$, $U_2 \in \mathbb{R}^{d_{in} \times r_2}$, and $U_3 \in \mathbb{R}^{d_{out} \times r_3}$. The factors admit natural interpretations in the MoE setting: U_1 forms r_1 meta-experts compressing inter-expert redundancy, while U_2 and U_3 define low-dimensional input and output subspaces. A Tucker-compressed MoE layer contains $r_1 r_2 r_3$ core parameters and $(K r_1 + d_{out} r_2 + d_{in} r_3)$ factor-matrix parameters, yielding $P_{\text{tucker}} = r_1 r_2 r_3 + (K r_1 + d_{out} r_2 + d_{in} r_3)$, compared to the original $P_{\text{orig}} = K\, d_{out} d_{in}$. This closed-form expression provides direct control over the desired compression ratio (Sec. 3.4). Overall, Tucker decomposition offers a unified mechanism that simultaneously compresses feature dimensions and removes cross-expert redundancy in a principled manner.

[1] The mode-n product is $(\mathcal{X} \times_n U)_{i_1 \ldots j \ldots i_3} = \sum_k \mathcal{X}_{i_1 \ldots k \ldots i_3} U_{kj}$, and the equivalent element-wise reconstruction is $\mathcal{T}_{w, i_1 i_2 i_3} = \sum_{r_1, r_2, r_3} \mathcal{G}_{r_1 r_2 r_3} U^{(1)}_{i_1 r_1} U^{(2)}_{i_2 r_2} U^{(3)}_{i_3 r_3}$.

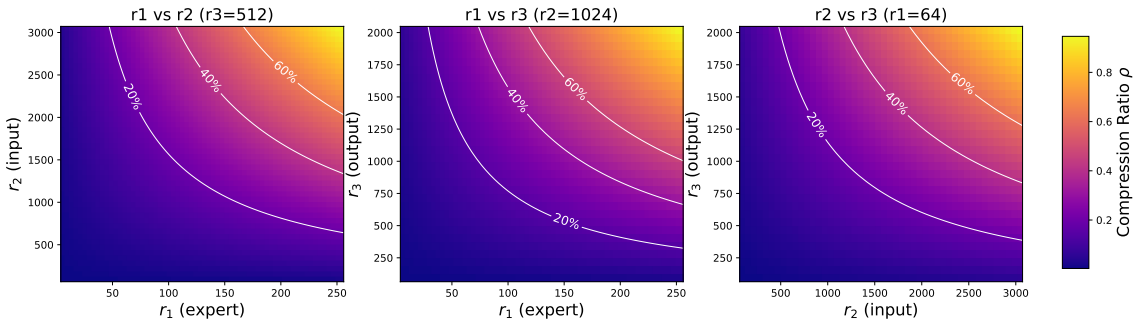


Figure 3: **Compression ratio (ρ) of Tucker-decomposed MoE layers.** Each subplot varies two Tucker ranks while fixing the third: (a) r_1 (expert) vs. r_2 (input) with $r_3 = 512$; (b) r_1 vs. r_3 (output) with $r_2 = 1024$; (c) r_2 vs. r_3 with $r_1 = 64$. White contours highlight $\rho = 20\%, 40\%, 60\%$.

3.3 Multi-Linear Tensor Whitening

Directly decomposing raw expert weights can be suboptimal because input activations are often highly correlated, making the feature space ill-conditioned (Yuan et al., 2023). As a result, the singular values of each $W^{(i)}$ do not faithfully reflect their contribution to the model's behavior, leading to inferior compression decisions. Prior work addresses this issue by applying whitening transformations that decorrelate features and rescale them to unit variance before decomposition (Wang et al., 2024; Li et al., 2025).

Multi-Linear Whitening. We extend the whitening to the tensorized MoE setting. Given a calibration dataset $\mathcal{D}_{\text{calib}}$, we collect both input activations $X \in \mathbb{R}^{N \times d_{in}}$ and output gradients $\nabla_Y \mathcal{L} \in \mathbb{R}^{N \times d_{out}}$, where N is the number of tokens. These statistics define the input and output covariances, $\Sigma_{in} = \frac{1}{N} X^T X$ and $\Sigma_{out} = \frac{1}{N} (\nabla_Y \mathcal{L})^T (\nabla_Y \mathcal{L})$. Here, Σ_{in} captures correlations among input features, while Σ_{out} reflects output sensitivities with respect to the loss. To obtain whitening transformations, we compute regularized square-root inverses of these covariances, namely $S_{in} = (\Sigma_{in} + \epsilon I)^{-\frac{1}{2}}$ and $S_{out} = (\Sigma_{out} + \epsilon I)^{-\frac{1}{2}}$, where ϵ is a small constant for numerical stability. Applying these to expert tensor $\mathcal{T}$ yields the whitened representation:

$$\mathcal{T}_w = \mathcal{T} \times_2 S_{in} \times_3 S_{out}. \tag{4}$$

This *multi-linear whitening* can decorrelate input or output modes, or both simultaneously, producing a well-conditioned tensor that improves the stability and effectiveness of Tucker decomposition. Unlike prior 2-D whitening approaches, our tensor-level whitening leverages multi-linear structure and explicitly adapts to input or output statistics, ensuring that TD-MoE not only captures cross-expert redundancy but also aligns the decomposition with the statistical geometry of data.

Re-coloring for Inference. To deploy compressed models efficiently, we absorb the inverse whitening transforms into the Tucker factors. Specifically, the original expert tensor is recovered as $\mathcal{T} \approx \mathcal{G} \times_1 U_1 \times_2 (S_{in}^{-1} U_2) \times_3 (S_{out}^{-1} U_3)$. For inference, we store the pre-colored factors $U'^{(1)} = U_1$, $U'^{(2)} = S_{in}^{-1} U_2$, and $U'^{(3)} = S_{out}^{-1} U_3$, ensuring that whitening and re-coloring introduce no extra runtime cost. We next address how to adaptively allocate Tucker ranks (r_1, r_2, r_3) to balance compression efficiency and model fidelity.

3.4 Adaptive 3D Rank Allocation

We consider an MoE weight tensor $\mathcal{W} \in \mathbb{R}^{K \times d_{out} \times d_{in}}$ with original parameter count P_{orig} and Tucker-decomposed size P_{tucker}. To satisfy a target compression ratio ρ^*, we introduce an adaptive three-dimensional

rank allocation scheme that searches over the Tucker rank triplet (r_1, r_2, r_3) under the constraint $P_{\text{tucker}} \approx (1-\rho^*)P_{\text{orig}}$. For candidate pair (r_1, r_2), the feasible r_3 is obtained in closed form from the budget equation:

$$r_3 = \frac{(1-\rho^*)\,P_{\text{orig}} - (Kr_1 + d_{out}r_2)}{r_1 r_2 + d_{in}}, \tag{5}$$

followed by projection onto the valid range $1 \le r_3 \le d_{in}$. This formulation reduces the original 3D search to a two-dimensional sweep over (r_1, r_2) with a one-dimensional closed-form update for r_3, enabling efficient exploration of the rank space under strict parameter budgets. We select the rank triplet that minimizes deviation from the target model size under integer and box constraints. By sweeping $r_1 \in [1, K]$ and $r_2 \in [1, d_{out}]$ and computing the corresponding r_3 from Eq. 5, this method identifies decompositions that tightly match the desired compression ratio ρ^*. Figure 3 visualizes the induced compression landscape, where each subplot varies two ranks while fixing the third. The colormap encodes the realized compression ratio, and contour lines mark target levels (20%, 40%, 60%), revealing how feasible configurations form structured regions in the rank space and showing how it systematically selects solutions lying on or near the desired contours, ensuring both feasibility and compression budget. Algorithmic details appear in Appendix 2.

4 Experiments

We conduct a comprehensive set of experiments to validate the effectiveness and robustness of our proposed TD-MoE method on a diverse set of modern MoE architectures and downstream tasks.

4.1 Experimental Setup

Models and Tasks. We evaluate TD-MoE on state-of-the-art MoE models with varying numbers of experts: Qwen2-57B-A14B Team et al. (2024) with 64 standard and 8 shared experts, Mixtral-8x7B (Jiang et al., 2024) with 8 experts, and Phi-3.5-MoE (Abdin et al., 2024) with 16 experts. Post-compression performance is assessed on a diverse set of tasks to ensure a comprehensive evaluation. We evaluate our method on 10 tasks, including commonsense reasoning and language modeling. For commonsense reasoning, we evaluate zero-shot accuracy on seven benchmarks: OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), MathQA (Amini et al., 2019), ARC-easy, and ARC-challenge (Clark et al., 2018). All zero-shot evaluations are conducted using the LM-Evaluation-Harness framework (Gao et al., 2021) to ensure reproducibility and fair comparison with prior work. For language modeling, we measure Perplexity (PPL) on WikiText-2 (Merity et al., 2017), Penn Treebank (PTB) (Marcus & Marcinkiewicz, 1994), and a held-out test set from C4 (Raffel et al., 2020).

Implementation Details. All experiments use a fixed calibration set of 256 WikiText-2 samples to compute whitening statistics, and all results are obtained under a strict post-training setup *without fine-tuning*. Tucker decomposition is implemented in PyTorch Paszke et al. (2019) and TensorLy Kossaifi et al. (2019), and covariance eigenvalues below 10^{-3} are clipped for stability. We evaluate our method in two whitening versions under two complementary truncation settings. Full 3-mode truncation is applied on Qwen2-57B-A14B, a large-capacity MoE with more than 64 experts per layer, in which the expert dimension is compressed to demonstrate the full form of TD-MoE. Expert-preserving truncation is applied on Mixtral-8×7B to ensure a fair comparison with MoE-SVD Li et al. (2025), whose formulation keeps the expert dimension fixed. Pruning and quantization are orthogonal to the decomposition method and are therefore omitted from the main comparison, and their combined performance is presented in the later experiment section.

4.2 Compression Performance Comparison

Commonsense Reasoning Performance. The right portion of table 1 reports accuracy on seven commonsense reasoning tasks across multiple compression ratios on two MoE models. Three consistent trends

Table 1: Perplexity and accuracy under budget–constrained compression. Top: **Qwen2-57B-A14B**. Bottom: **Mixtral-8×7B**. Lower perplexity and higher accuracy indicate better performance.

Ratio	Method	Wiki.	PTB	C4	Openb.	ARC_e	WinoG.	HellaS.	ARC_c	PIQA	MathQA	Avg.
	Qwen2-57B-A14B (8+64 experts, 8+top-8 activated))											
0%	Original	5.12	9.18	8.86	0.33	0.75	0.74	0.63	0.46	0.81	0.39	0.59
20%	NAEE (2024)	6.96	10.39	10.52	0.31	0.72	0.72	0.59	0.42	0.79	0.36	0.56
	MoE-SVD (2025)	**5.41**	13.26	11.63	0.30	0.73	0.73	**0.61**	0.45	0.78	0.35	0.56
	TD-MoE (input)	6.65	10.40	**10.34**	0.30	**0.79**	**0.73**	0.58	**0.49**	0.80	**0.37**	**0.58**(↑4%)
	TD-MoE (output)	6.67	**10.28**	10.35	**0.31**	**0.79**	**0.73**	0.59	**0.49**	**0.80**	**0.37**	**0.58**(↑4%)
40%	NAEE (2024)	8.79	12.64	13.26	0.28	0.72	0.71	0.54	0.39	0.76	0.32	0.53
	MoE-SVD (2025)	13.43	23.88	18.83	0.29	0.63	0.65	0.45	0.33	0.71	0.31	0.48
	TD-MoE (input)	**7.96**	**12.37**	**12.91**	**0.31**	**0.77**	**0.71**	**0.54**	0.44	**0.78**	**0.36**	**0.56**(↑6%)
	TD-MoE (output)	8.07	12.50	13.09	0.29	**0.77**	**0.71**	**0.54**	**0.45**	**0.78**	0.35	**0.56**(↑6%)
60%	NAEE (2024)	14.79	23.52	25.00	0.21	0.58	0.61	0.44	0.29	0.69	0.26	0.44
	MoE-SVD (2025)	19.46	29.94	25.06	0.27	0.62	0.64	0.44	0.32	0.69	0.30	0.47
	TD-MoE (input)	**12.21**	**19.87**	**20.49**	**0.28**	**0.73**	**0.69**	0.45	**0.41**	**0.75**	**0.31**	**0.52**(↑11%)
	TD-MoE (output)	12.49	19.92	21.04	**0.28**	**0.73**	0.68	**0.46**	**0.41**	0.74	**0.31**	**0.51**(↑9%)
	Mixtral-8×7B (8 experts, top-2 activated)											
0%	Original	3.84	14.70	7.18	0.35	0.84	0.76	0.65	0.57	0.82	0.43	0.63
20%	NAEE (2024)	4.77	16.09	8.89	0.32	0.76	0.72	0.58	0.47	0.79	0.40	0.58
	MoE-SVD (2025)	4.86	19.42	8.98	0.33	0.79	0.74	0.56	0.49	0.78	0.37	0.58
	TD-MoE (input)	4.67	19.82	8.04	0.32	0.82	0.76	0.61	0.53	0.82	0.40	0.61 (↑5%)
	TD-MoE (output)	**4.49**	17.20	**7.73**	**0.33**	**0.83**	**0.77**	**0.64**	**0.53**	**0.82**	**0.40**	**0.62**(↑7%)
40%	NAEE (2024)	6.44	22.15	13.86	0.25	0.63	0.64	0.46	0.36	0.72	0.35	0.48
	MoE-SVD (2025)	6.74	27.73	12.41	0.27	0.72	0.67	0.43	0.38	0.71	0.32	0.50
	TD-MoE (input)	7.17	32.10	11.44	0.28	0.76	0.72	0.49	0.43	0.78	0.34	0.54 (↑8%)
	TD-MoE (output)	**5.79**	24.60	**9.21**	**0.28**	**0.77**	**0.76**	**0.57**	**0.47**	**0.79**	**0.35**	**0.57**(↑14%)
60%	NAEE (2024)	11.43	47.28	31.16	0.17	0.42	0.55	0.33	0.23	0.62	0.26	0.36
	MoE-SVD (2025)	13.52	130.26	39.54	0.19	0.45	0.55	0.33	0.23	0.62	0.25	0.37
	TD-MoE (input)	13.64	66.00	**19.63**	**0.22**	**0.58**	**0.63**	**0.40**	**0.34**	**0.73**	**0.27**	**0.45** (↑22%)
	TD-MoE (output)	17.78	79.43	**24.85**	0.21	0.55	0.62	0.38	0.28	0.65	0.24	**0.42** (↑14%)

emerge. (a) At the 20% compression level, TD-MoE preserves accuracy remarkably well on both models, with less than 1% absolute drop relative to the original model. Both input- and output-whitening variants outperform strong baselines, and the output-whitening variant achieves improvements of up to four points (4% and 7% relative gains). Notably, the gains are most pronounced on reasoning-intensive tasks such as HellaSwag and ARC-c, which are highly sensitive to expert representation quality. (b) At the 40% compression level, TD-MoE remains robust. The output-whitening variant reaches 0.56 on Qwen2-57B-A14B and 0.57 on Mixtral-8×7B, yielding 17% and 14% relative improvements over MoE-SVD. The input-whitening variant shows similarly stable behavior, demonstrating the regularization and conditioning benefits of whitening during Tucker truncation. (c) At the 60% compression level, performance degradation is unavoidable across all methods due to the aggressive parameter reduction. However, TD-MoE remains substantially more resilient: on both models it delivers large margins of 11% and 21.6% relative improvement over MoE-SVD. These results indicate that whitening-based Tucker decomposition preserves structural information more effectively than these baselines, especially under high compression. Detailed configurations are provided in the supplementary material, and additional experiments on Phi-3.5-MoE are included in Appendix A.5.

Language Modeling Perplexity Comparison. The left portion of Table 1 reports perplexity on WikiText-2, PTB, and C4 for Qwen2-57B-A14B and Mixtral-8×7B across multiple compression ratios. At 20% compression, the output-whitening variant attains strong performance (e.g., 4.49/17.20/7.73 on Mixtral-8×7B), clearly outperforming NAEE and MoE-SVD, while the input-whitening variant remains competitive. At 40% compression, baselines degrade substantially, whereas our output-whitening model maintains low perplexity (5.79/24.60/9.21), showing clear advantages. At 60% compression, perplexity rises for all methods, but TD-MoE degrades more gracefully. We also observe that output whitening performs slightly better at

lower compression, while input whitening becomes more stable under more aggressive truncation. This observation is consistent with the role of whitening in controlling error propagation: at lower compression, output whitening more effectively preserves the dominant directions of the output distribution, while at higher compression, input whitening stabilizes the more severe rank-reduction by preventing amplification of poorly conditioned input activations. These results demonstrate that whitening-based Tucker decomposition provides a consistent advantage over strong baselines across compression levels.

4.3 Ablation Studies

Analysis of Whitening. We analyze how multi-linear whitening affects the numerical structure of activations and the resulting Tucker decomposition. Beyond performance comparison (Table 1), we quantify whitening's impact on covariance spectra and cross-dimensional correlations. (a) Before whitening, activation covariances exhibit extremely ill-conditioned spectra, with eigenvalues spanning 9-5.4×10^4 across Mixtral-8×7B layers. After whitening, all eigenvalues collapse to 1.0 with maximum deviations below 10^{-7} (rightmost column in Table 2), removing scale anisotropy and stabilizing truncated Tucker factors. (b) Off-diagonal correlations reach 0.63–0.79 pre-whitening with standard deviations around 10^{-2}; whitening eliminates these entirely, reducing all residual correlations to below 10^{-7} and producing fully decorrelated activation subspaces. These numerical improvements translate into consistent empirical gains across datasets, with both input and output whitening yielding noticeably more stable performance under compression.

Table 2: Covariance spectra and cross-dimensional correlations before and after whitening.

Layer	$\lambda_{\min}$–$\lambda_{\max}$ (Pre)	Corr Std / Max (Pre)	Post-Whitening Deviation
3	22 – 3.9×10^4	1.06×10^{-2} / 0.72	1.0×10^{-7}
5	31 – 3.8×10^4	1.14×10^{-2} / 0.64	3.0×10^{-8}
7	18 – 4.3×10^4	1.27×10^{-2} / 0.78	6.0×10^{-8}
9	9 – 5.4×10^4	1.42×10^{-2} / 0.63	1.1×10^{-7}
12	10 – 5.3×10^4	1.34×10^{-2} / 0.76	1.0×10^{-7}
24	24 – 1.9×10^4	1.01×10^{-2} / 0.79	4.0×10^{-8}

Complexity and Runtime Analysis. We analyze the complexity and quantify the computational and memory overhead of whitening process, Cholesky decomposition, and Tucker decomposition in TD-MoE. (a) Whitening introduces less than 1% additional memory usage and only a small compute cost (0.11 TFLOPs). Since it is applied once per layer to a 2D covariance matrix, its cost does not grow with the number of experts. All intermediate tensors are released immediately after each layer, keeping the peak memory footprint low. (b) Cholesky decomposition scales reliably to large FFN dimensions (up to 14,336) and is not a practical bottleneck. Table 3 reports the measured overhead on NVIDIA A800 GPUs. (c) For decomposition, Tucker exhibits asymptotically comparable or lower computational cost than per-expert SVD. While per-expert SVD requires $E\cdot O(MN\min(M,N))$ work for E experts, Tucker (rand-based) performs only three SVDs on mode-k unfoldings and benefits from randomized SVD, reducing each step to $O(MNr)$. As shown in Table 3, Tucker remains within a small constant factor of per-expert SVD on Mixtral-8×7B and becomes faster at higher compression ratios; on Phi-3.5-MoE with 16 experts, it is consistently 2.5–4.6× faster due to the larger expert count. All computations are performed offline, incuring no additional overhead at inference.

Table 3: Overhead and runtime.

Component	**Mixtral-8×7B**	**Phi-3.5-MoE**
Whitening Mem.	890 MB	231 MB
Whitening FLOPs	0.11 TFLOPs	0.02 TFLOPs
Cholesky (max dims)	784 MB	156 MB
Tucker (rand) / SVD	20.7 s / 9.4 s	7.1 s / 17.7 s
Relative Speed	1.3–2.2× slower	2.5–4.6× faster

Compatibility with Quantization and Pruning. We examine whether TD-MoE composes effectively with common post-training compression techniques. Two settings are considered on Mixtral-8×7B: (i) applying 8-bit NF4 quantization Dettmers et al. (2023) after TD-MoE, and (ii) structured pruning that removes low-energy core slices together with their corresponding factor-matrix columns. As shown in Table 4, 8-bit quantization introduces virtually no additional degradation: at both 20% and 40% compression, LM averages shift by no more than 0.1, and reasoning accuracy (OpenbookQA, Arc-E, Wino, Arc-C, PIQA) remains unchanged. Structured pruning follows a smooth, monotonic degradation pattern: under 20% compression, increasing sparsity from 0% to 40% gradually reduces LM Avg from 10.85 to 15.72 and Reason Avg from 0.66 to 0.61, with similar behavior observed under 40% compression. These results indicate that TD-MoE is compatible with both quantization and pruning. The pruning procedure operates directly in the Tucker domain by ranking and removing low-energy core slices and their associated factor-matrix columns; detailed pseudocode is provided in the appendix A.4.

Table 4: TD-MoE combined with 8-bit NF4 quantization and structured pruning (Mixtral-8×7B).

Setting	Prune	LM Avg	Rea. Avg
Original	–	7.92	0.63
20% Compression			
TD-MoE	0%	10.85	0.66
+ 8-bit NF4	0%	10.69	0.66
+ Pruning	20%	12.72	0.64
+ Pruning	30%	14.21	0.62
+ Pruning	40%	15.72	0.61
40% Compression			
TD-MoE	0%	15.20	0.57
+ 8-bit NF4	0%	15.16	0.56
+ Pruning	40%	30.06	0.54

Sensitivity to Calibration Size and Clipping Threshold. We assess the impact of two whitening-related hyperparameters: the calibration set size N and the eigenvalue clipping threshold τ. As shown in Table 5, varying N from 128 to 2K (and switching the calibration corpus from WikiText-2 to PTB) changes average perplexity by no more than 0.03 and the average accuracy across downstream tasks (ARC-E, WinoG, ARC-C, PIQA) by at most 0.01. Similarly, sweeping τ across four orders of magnitude (10^{-1}–10^{-4}) under both 20% and 40% compression yields fluctuations no larger than 0.01 in accuracy and below 0.1 in perplexity. Notably, the relative ranking of compression settings remains unchanged across all configurations, suggesting that whitening behaves in a stable and well-conditioned regime. These results indicate that the whitening process is numerically robust and largely insensitive to calibration size, corpus choice, or clipping threshold.

Table 5: Sensitivity of TD-MoE to calibration size N and clipping threshold τ.

Setting	Avg PPL	Avg Acc	Variation
$N = \{128, 256, 512, 1\text{k}, 2\text{k}\}$	9.81–9.84	0.73–0.74	$\Delta\text{PPL} \leq 0.03$, $\Delta\text{Acc} \leq 0.01$
$\tau \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ (20%)	7.23–7.32	0.73–0.74	$\Delta\text{PPL} \leq 0.09$, $\Delta\text{Acc} \leq 0.01$
$\tau \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ (40%)	10.13–10.20	0.69–0.70	$\Delta\text{PPL} \leq 0.07$, $\Delta\text{Acc} \leq 0.01$

5 Conclusion

In this paper, we introduced TD-MoE, a cross-expert tensor decomposition method that rethinks MoE compression from a global perspective. By unifying expert weights into a 3D tensor, incorporating robust multilinear whitening, and allocating ranks through principled budget-aware strategies, our approach offers a scalable method for jointly reducing cross-and intra-expert redundancy while preserving model fidelity. Extensive experiments demonstrate that TD-MoE achieves competitive or superior accuracy under substantial compression ratios and composes with post-training quantization and pruning. Beyond demonstrating its effectiveness on MoE models, this work highlights the broader potential of the tensor-based compression method as a foundation for more efficient, adaptable, and widely deployable large-scale language models.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. This work does not raise any specific ethical concerns. It focuses on model compression techniques for Mixture-of-Experts models and does not involve sensitive data, human subjects, or high-risk applications. All datasets used, including , OpenbookQA, WinoGrande, HellaSwag, PIQA, MathQA, ARC-easy, a, ARC-challenge, WikiText-2, PTB, and C4 are sourced in compliance with relevant usage guidelines, ensuring no violation of privacy.

REPRODUCIBILITY STATEMENT

We ensure reproducibility by providing detailed descriptions of our methods, datasets, and experimental settings in the main text. The codebase for reproducing all results has been submitted via an anonymous URL and will be released publicly upon publication.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of NAACL-HLT*, pp. 2357–2367, 2019.

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

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

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

Hao Gu, Wei Li, Lujun Li, Zhu Qiyuan, Mark G Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for moe-based llms compression. In *Forty-second International Conference on Machine Learning*, 2025.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. *arXiv preprint arXiv:2505.03804*, 2025.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and XIAOJUAN QI. Mixture compressor for mixture-of-experts llms gains more. In *The Thirteenth International Conference on Learning Representations*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

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

Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learnin python. *Journal of Machine Learning Research*, 20(26):1–6, 2019.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *The Twelfth International Conference on Learning Representations*, 2024.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G Lee, Shengjie Sun, Wei Xue, and Yike Guo. Moe-svd: Structured mixture-of-experts llms compression via singular value decomposition. In *Forty-second International Conference on Machine Learning*, 2025.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *ACL (1)*, 2024.

Mitchell P Marcus and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2), 1994.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision*, pp. 522–539. Springer, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2024.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. Moe-i2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10456–10466, 2024.

Miao Yin, Huy Phan, Xiao Zang, Siyu Liao, and Bo Yuan. Batude: Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8874–8882, 2022.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Peining Zhen, Ziyang Gao, Tianshu Hou, Yuan Cheng, and Hai-Bao Chen. Deeply tensor compressed transformers for end-to-end object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4716–4724, 2022.

Yuhang Zhou, Giannis Karamanolakis, Victor Soto, Anna Rumshisky, Mayank Kulkarni, Furong Huang, Wei Ai, Jianhua Lu, and AGI Amazon. Mergeme: Model merging techniques for homogeneous and heterogeneous moes. 2025.

A APPENDIX

A.1 USE OF LLMS

LLMs were used for related work surveys and grammatical refinement of the paper.

A.2 DETAILS OF THE METHOD

Algorithm 1 Tensor Decomposition for Mixture-of-Experts (TD-MoE)

Require: Original expert weights $\{W^{(i)} \in \mathbb{R}^{d_{in} \times d_{out}}\}_{i=1}^{K}$; Calibration dataset $\mathcal{D}_{\text{calib}}$; Target compression ratio ρ; Regularization constant ϵ.

Ensure: Compressed core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$; Re-colored factor matrices $U'^{(1)}, U'^{(2)}, U'^{(3)}$.

// Step 1: Cross-Expert Tensorization

Initialize a tensor $\mathcal{T} \in \mathbb{R}^{K \times d_{in} \times d_{out}}$.

for $i = 1$ to K **do**

  $\mathcal{T}[i, :, :] \leftarrow W^{(i)}$ ▷ Stack expert weights into a 3D tensor

end for

// Step 2: Collect Statistics for Whitening

$X, \nabla_Y \mathcal{L} \leftarrow \text{CollectActivationsAndGradients}(\{W^{(i)}\}, \mathcal{D}_{\text{calib}})$

$\Sigma_{in} \leftarrow \frac{1}{N} X^T X$ ▷ Compute input covariance

$\Sigma_{out} \leftarrow \frac{1}{N} (\nabla_Y \mathcal{L})^T (\nabla_Y \mathcal{L})$ ▷ Compute output covariance

// Step 3: Multi-Linear Tensor Whitening

$S_{in} \leftarrow (\Sigma_{in} + \epsilon I)^{-\frac{1}{2}}$ ▷ Compute input whitening matrix via Cholesky

$S_{out} \leftarrow (\Sigma_{out} + \epsilon I)^{-\frac{1}{2}}$ ▷ Compute output whitening matrix

$\mathcal{T}_w \leftarrow \mathcal{T} \times_2 S_{in} \times_3 S_{out}$ ▷ Apply whitening to input/output modes

// Step 4: Adaptive 3D Rank Allocation

$(r_1, r_2, r_3) \leftarrow \text{FindOptimalRanks}(\rho, K, d_{in}, d_{out})$ ▷ Search for ranks satisfying the budget

// Step 5: Joint Tucker Factorization

$\mathcal{G}, \{U_1, U_2, U_3\} \leftarrow \text{TuckerDecomposition}(\mathcal{T}_w, \text{ranks} = (r_1, r_2, r_3))$

// Step 6: Re-coloring for Inference

$S_{in}^{-1} \leftarrow (\Sigma_{in} + \epsilon I)^{\frac{1}{2}}$

$S_{out}^{-1} \leftarrow (\Sigma_{out} + \epsilon I)^{\frac{1}{2}}$

$U'^{(1)} \leftarrow U_1$

$U'^{(2)} \leftarrow S_{in}^{-1} U_2$ ▷ Absorb inverse whitening into input factor

$U'^{(3)} \leftarrow S_{out}^{-1} U_3$ ▷ Absorb inverse whitening into output factor

return $\mathcal{G}, U'^{(1)}, U'^{(2)}, U'^{(3)}$

A.3 BUDGET-CONSTRAINED RANK SEARCH (BCRS).

The most direct approach is to enforce the global parameter budget as closely as possible. For a given candidate r_1, we solve the budget equation for the feasible r_2:

$$r_3 = \frac{(1 - \rho^*) P_{\text{orig}} - (K r_1 + d_{out} r_2)}{r_1 r_2 + d_{in}}.$$

We then iterate over $r_1 \in [1, d_{out}]$, compute the corresponding r_2, and retain only feasible values with $1 \le r_2 \le d_{in}$. Among all such candidates, the pair minimizing

$$\Delta(r_1, r_2) = \left| P_{\text{tucker}}(r_1, r_2) - (1 - \rho^*) P_{\text{orig}} \right|$$

is selected. BCRS guarantees that the compressed parameter count is tightly aligned with the target budget, but the resulting ranks may be unbalanced across modes.

Algorithm 2 Budget-Constrained Rank Search (BCRS)

Require: MoE tensor size $(K, d_{\text{out}}, d_{\text{in}})$; target compression ratio $\rho^* \in (0, 1)$
Ensure: Tucker ranks (r_1, r_2, r_3)

$P_{\text{orig}} \leftarrow K\, d_{\text{out}} d_{\text{in}}$
$P_{\text{target}} \leftarrow (1 - \rho^*) P_{\text{orig}}$
best_err $\leftarrow +\infty$, $(r_1^*, r_2^*, r_3^*) \leftarrow (1, 1, 1)$
for $r_1 = 1, 2, \ldots, K$ **do**
  for $r_2 = 1, 2, \ldots, d_{\text{out}}$ **do**
    num $\leftarrow P_{\text{target}} - (K r_1 + d_{\text{out}} r_2)$
    den $\leftarrow r_1 r_2 + d_{\text{in}}$
    if den ≤ 0 **then**
      continue
    end if
    $r_3 \leftarrow \lfloor \text{num}/\text{den} \rfloor$
    if $r_3 < 1$ **or** $r_3 > d_{\text{in}}$ **then**
      continue
    end if
    $P_{\text{tucker}} \leftarrow r_1 r_2 r_3 + (K r_1 + d_{\text{out}} r_2 + d_{\text{in}} r_3)$
    err $\leftarrow \left| P_{\text{tucker}} - P_{\text{target}} \right|$
    if err $<$ best_err **then**
      best_err $\leftarrow$ err
      $(r_1^*, r_2^*, r_3^*) \leftarrow (r_1, r_2, r_3)$
    end if
  end for
end for
return (r_1^*, r_2^*, r_3^*)

For layer selection, we follow MoE-SVD Li et al. (2025) and allocate larger ranks to layers with higher expert-activation frequency, reflecting their greater functional contribution during routing. We additionally provide a Fisher-regularized variant with optional smoothing to further stabilize layerwise allocation and avoid abrupt fluctuations across adjacent layers. In practice, combining activation frequency with a smoothed Fisher prior yields a simple, low-overhead mechanism for distributing the global compression budget while respecting layerwise sensitivity as $P_\ell^* = \widehat{w}_\ell (1 - \rho^*) \sum_{\ell=1}^{L} P_{\text{orig}}^{(\ell)}$. A visual illustration is provided in Fig. 4.

A.4 Structured Tucker Pruning

Below we describe the post-training structured pruning applied after Tucker decomposition to remove redundant latent dimensions. Starting from a Tucker model with core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and factor matrices $\{U_1, U_2, U_3\}$, we operate entirely in the latent space: no access to the original dense weights is required. The key idea is to exploit the energy distribution inside the core tensor. Each mode-2 slice $\mathcal{G}_{:,i,:}$ corresponds to one latent feature direction in the output mode, and each mode-3 slice $\mathcal{G}_{:,:,j}$ corresponds to one latent direction in the input mode. We measure the contribution of each slice via its ℓ_2 norm (energy), rank slices

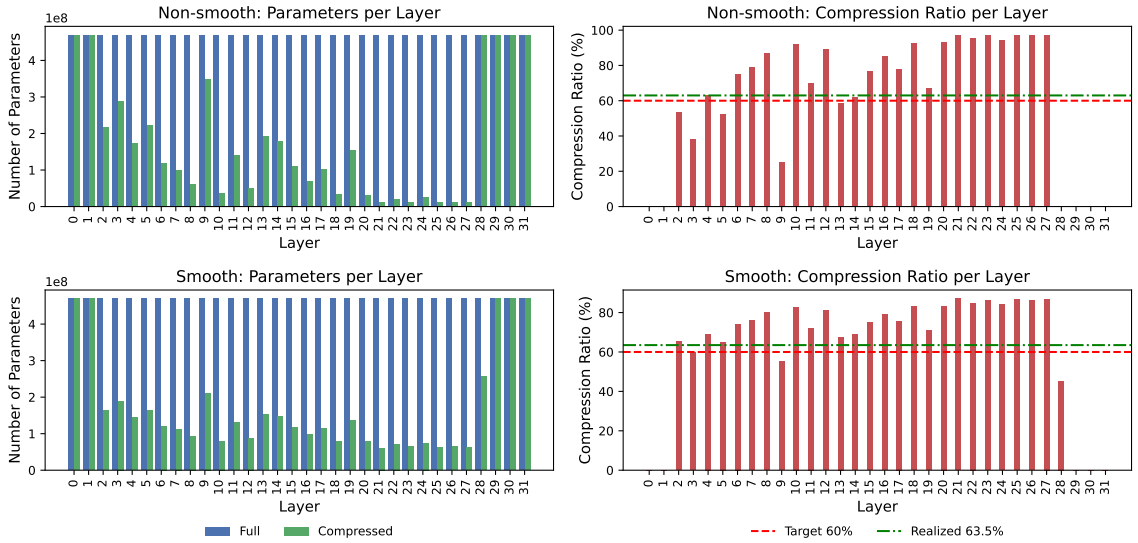


Figure 4: Layerwise Fisher-regularized rank allocation with and without smoothing under 60% compression.

by energy, and treat the lowest energy slices as redundant. Low-energy slices contribute little to the overall reconstruction and can therefore be safely removed, together with their associated columns in the factor matrices U_2 and U_3. Concretely, given a pruning ratio ρ, we retain only the top $(1-\rho)$ fraction of slices along modes 2 and 3, and shrink the core and factor matrices accordingly. This yields a strictly smaller Tucker model with reduced ranks (r_2', r_3'), while preserving the multilinear structure and the original expert mode U_1. Because pruning is performed on top of an already compressed decomposition, it can be applied incrementally to trade parameters for accuracy with smooth, monotonic degradation, and does not require re-running Tucker. The full pruning pipeline is summarized in Algorithm 3.

Algorithm 3 Structured Tucker Pruning for TD-MoE

Require: Core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, factor matrices $\{U_1, U_2, U_3\}$, pruning ratio $\rho \in [0, 1)$
Ensure: Pruned core $\mathcal{G}'$ and factors $\{U_1', U_2', U_3'\}$

if $\rho \leq 0$ **then**
  return $\mathcal{G}, \{U_1, U_2, U_3\}$
end if
$r_2' \leftarrow \max(1, \lfloor (1-\rho) r_2 \rfloor), \quad r_3' \leftarrow \max(1, \lfloor (1-\rho) r_3 \rfloor)$
$e^{(2)} \leftarrow \|\mathcal{G}\|_{(1,3)} \in \mathbb{R}^{r_2}, \quad I_2 \leftarrow \text{TopK}(e^{(2)}, r_2')$
$e^{(3)} \leftarrow \|\mathcal{G}\|_{(1,2)} \in \mathbb{R}^{r_3}, \quad I_3 \leftarrow \text{TopK}(e^{(3)}, r_3')$
$\mathcal{G}' \leftarrow \mathcal{G}[:, I_2, I_3]$
$U_1' \leftarrow U_1, \quad U_2' \leftarrow U_2[:, I_2], \quad U_3' \leftarrow U_3[:, I_3]$
return $\mathcal{G}', \{U_1', U_2', U_3'\}$

A.5 CONFIGURATION ANALYSIS.

Table 6 summarizes the effect of different configuration choices on Qwen2-57B-A14B, including (i) which MoE layers are selected for compression, (ii) how many experts are retained in those layers, and (iii) the resulting layer-wise compression ratios. *Reason Avg* denotes the averaged accuracy over reasoning benchmarks (OpenBookQA, ARC_e, WinoG, ARC_c, PIQA). Across all ratios, TD-MoE demonstrates a clear advantage: moderate adjustments to layer selection and expert retention lead to substantial improvements over MoE-SVD, with the best-performing setups highlighted in **bold**. Notably, even under heavy compression (40–60%), TD-MoE maintains strong stability compared with baselines.

Ratio	Method	Layer Sel.	Retain Exp.	Layer Ratio	Wiki2	PTB	C4	Reason Avg
origin	-	28	64	100%	5.87	10.87	9.14	0.626
20%	MoE-SVD (2025)	*	64	*	5.41	13.26	11.63	0.600
	TD-MoE	6	8	0.20	6.98	10.70	10.73	0.612
	TD-MoE	**8**	**16**	**0.20**	6.75	10.60	10.85	**0.626**
40%	MoE-SVD (2025)	*	64	*	13.43	23.88	18.83	0.522
	TD-MoE	**16**	**16**	**0.20**	8.53	13.73	14.12	**0.614**
	TD-MoE	14	8	0.40	8.01	12.39	12.90	0.602
60%	MoE-SVD (2025)	*	64	*	19.46	29.94	25.06	0.520
	TD-MoE	**20**	**8**	**0.60**	12.40	19.88	20.96	**0.568**

Table 6: Performance comparison across different configurations on Qwen2-57B-A14B.

A.6 ADDITIONAL RESULTS ON PHI-3.5-MOE

To further evaluate robustness, Table 7 presents results on Phi-3.5-MoE across 20%, 40%, and 60% compression. TD-MoE consistently ranks among the strongest methods. At 20–40% compression, TD-MoE closely matches the original model on language modeling metrics and achieves the highest reasoning accuracy among all baselines. Even under aggressive 60% compression—a regime where most decompositions collapse—TD-MoE remains significantly more stable and preserves far better accuracy, highlighting its resilience under extreme compression.

Ratio	Method	W2	PTB	C4	Reason Avg
origin	baseline	3.5	8.4	8.2	0.62
20%	ASVD Yuan et al. (2023)	7.2	10.7	9.6	0.56
	SVD-LLM Wang et al. (2024)	8.3	14.8	12.9	0.51
	MoE-SVD Li et al. (2025)	4.6	10.1	9.9	0.60
	TD-MoE	4.7	9.2	9.1	0.61
40%	MoE-I2 Yang et al. (2024)	7.5	21.0	21.0	0.45
	SVD-LLM Wang et al. (2024)	38.8	68.5	43.8	0.43
	MoE-SVD Li et al. (2025)	5.5	11.7	11.9	0.56
	TD-MoE	6.4	10.9	10.8	0.58
60%	ASVD Yuan et al. (2023)	107.7	208	161	0.35
	SVD-LLM Wang et al. (2024)	7168	7101	7119	0.31
	MoE-SVD Li et al. (2025)	7.5	21.0	21.9	0.49
	TD-MoE	13.7	19.7	17.9	0.50

Table 7: Performance comparison on Phi-3.5-MoE.

A.7 Additional Results on HumanEval and XNLI

To broaden our evaluation, we further include code generation (HumanEval) and multilingual NLI (XNLI). Table 8 reports results on Mixtral-8×7B under 0.2 and 0.4 compression budgets. TD-MoE consistently surpasses MoE-SVD on both tasks. On HumanEval, TD-MoE improves Pass@1 by +0.161 and +0.103 at 0.2 and 0.4 compression, whereas MoE-SVD collapses due to independently truncating each expert, which destroys shared code-related subspaces and magnifies decomposition errors. On XNLI, TD-MoE also maintains higher accuracy at both compression levels, preserving multilingual reasoning ability more effectively than per-expert SVD. These results further demonstrate the robustness and generality of TD-MoE across both code-generation and multilingual inference tasks.

Table 8: Results on Mixtral-8×7B for HumanEval (Pass@1) and XNLI.

Task	Ratio	MoE-SVD	TD-MoE	Δ
HumanEval (Pass@1)	Origin	–	0.293	–
HumanEval (Pass@1)	0.2	0.079	0.240	**+0.161**
HumanEval (Pass@1)	0.4	0.037	0.140	**+0.103**
XNLI (Acc.)	Origin	–	0.450	–
XNLI (Acc.)	0.2	0.390	0.440	**+0.050**
XNLI (Acc.)	0.4	0.380	0.410	**+0.030**