Structure, Subspace and System: Push the Real Limit of Extremely Low-Bit Quantization for MoE-LLMs

Jiaqi Zhao[*], **Zichen Li**[*], **Yixuan Dong, Miao Zhang**[†], **Weili Guan, Liqiang Nie**
Harbin Institute of Technology (Shenzhen)
jiaqizhao0455@outlook.com {2023110630,24S151010}@stu.hit.edu.cn
{zhangmiao,guanweili}@hit.edu.cn nieliqiang@gmail.com

Abstract

Mixture-of-Experts (MoE) large language models (LLMs) suffer significant inference and storage overheads, which makes extremely low-bit quantization (sub 2-bit) highly desirable. However, existing methods for dense LLMs typically quantize each expert independently and require it to approximate the original mapping over the entire input space. We argue that this formulation is overly conservative and structurally mismatched for MoE: **1)** experts in the same MoE layer are not fully independent but operate on a shared hidden representation space. Under low-bit budgets, independently binarizing each expert will distort the activation-sensitive input geometry, which leads to severe performance degradation, and **2)** due to routing-induced specialization, each expert only serves a concentrated subset of tokens, whose activations typically occupy a much narrower subspace than that of dense layers. More importantly, when the weights approach 1-bit, the dominant deployment bottleneck is no longer the weights themselves, but the metadata (e.g., scaling factors, bitmap, grouping index) which can raise the real effective inference precision to around 3-bit. This issue, however, has largely been overlooked in prior works. Building on these perspectives, we introduce a novel extremely low-bit quantization framework for MoE-LLMs, called **TriS-MoE**, which considers *Structure*, *Subspace* and *System*. Specifically, we first extract a shared high-precision input-side backbone to preserve activation-sensitive input geometry, while binarizing the remaining components to save costs. Secondly, we propose to redirect expert quantization errors into the null space of routed activations to minimize output perturbation on the subspace actually used by each expert. Finally, we co-design a bitmap compression method based on Golomb-Rice coding which approaches the Shannon entropy lower bound and a specialized streaming loading mechanism to reduce metadata overhead during inference. Extensive experiments on MoE-LLMs demonstrate our **TriS-MoE** outperforms the strongest baseline by 8.43% average accuracies, and also achieves an average $6.32\times$ reduction in inference memory on real systems, enabling Qwen3.5-35B-A3B and Mixtral-8$\times$7B to run on a single consumer-GPU. Codes are available at https://github.com/iLearn-Lab/NeurIPS26-TriS-MoE.

1 Introduction

Mixture-of-Experts (MoE) [1, 2, 3, 4] architecture efficiently scales up the parameters for large language models (LLMs) [5, 6, 7] by replacing MLP blocks with multiple specialized expert networks, which has demonstrated remarkable success. However, the large collection of expert parameters

[*]Equal contribution.
[†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

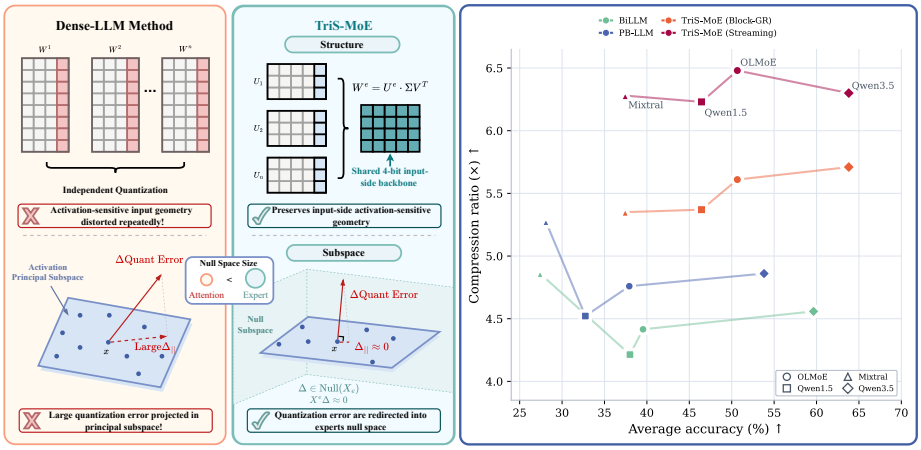


Figure 1: Directly extending dense-LLMs PTQ methods to MoE-LLMs is suboptimal. Our **TriS-MoE** achieves real extremely low-bit inference with structure, subspace and system level design.

introduces unavoidable storage and inference overheads. For example, Mixtral-8×7B [8] requires over 90 GiB memory for inference, which exceeds the capacity of an NVIDIA A100-80GB GPU. Such severe deployment challenge highlights the urgent need for effective model compression [9, 10, 11].

Quantization [12, 13, 14, 15] offers a promising solution to this issue, especially for extremely low-bit post-training (PTQ) quantization [16, 17] which can efficiently compress the weights to sub 2-bit, so it is particularly appealing for MoE-LLMs. However, existing low-bit PTQ methods [18, 19, 20] are primarily designed for dense LLMs and do not adequately consider the unique characteristics of MoE. Specifically, we discover two mismatches when extending dense-specialized methods to MoE-LLMs: ***Firstly***, they typically quantize each expert independently, which ignores a crucial structural property of MoE: experts in the same layer all operate on hidden states drawn from the same upstream representation space. Since weights corresponding to salient input activations are sensitive to quantization errors [21, 22, 23], directly binarizing each expert independently repeatedly distorts the activation-sensitive input geometry and will cause significant performance degradation. ***Secondly***, they require each quantized expert to approximate the original mapping over the entire input space, which is too difficult for low-bit quantization and overly conservative because each expert only serves a concentrated subset of routed tokens whose activations occupy a much narrower subspace. Therefore, we argue that extremely low-bit PTQ for MoE-LLMs should not be viewed as a direct extension of dense LLMs quantization, but as a problem that requires MoE-specific reformulation.

In addition to these mismatches, another more important practical bottleneck emerges which has been ignored by existing works. When the weights approach 1-bit, the actual deployment cost is no longer determined by the main weights alone, but increasingly by the quantization metadata [24, 25], such as the scaling factors, bitmaps, and grouping index, which can dominate the effective storage and execution overhead. Specifically, to mitigate the severe accuracy degradation at low-bit, existing methods often rely on compensation strategies such as fine-grained grouping and preserving unstructured salient weights, which substantially increase metadata overhead. For instance, PB-LLM [19] uses 1.7-bit weights but requires an additional 0.27-bit for scaling factors and 1-bit for grouping bitmaps, while BiLLM [20] achieves 1.1-bit weights but incurs extra 1-bit for both of them, respectively (refer to Appendix C). As a result, although these methods achieve very low weight precision, but still fall short of real extremely low-bit deployment in practice.

Guided by these observations, we propose a novel extremely low-bit PTQ method for MoE-LLMs, called **TriS-MoE**, from the perspectives of *structure*, *subspace* and *system*, as shown in Figure 1. Specifically, at the structure level, **TriS-MoE** vertically concatenates expert weights in each MoE layer and performs SVD to extract a shared input-side backbone. We preserve this backbone in 4-bit to protect the activation-sensitive input geometry of the layer, while binarizing the remaining components to reduce storage cost. Then at the subspace level, inspired by [26] we introduce to redirect expert quantization errors into the null space of routed activations to minimize output perturbation on the principal subspace actually used by each expert. To enable accurate and adaptive

null space approximation, we propose to leverage Kneedle algorithm [27] for the discrete eigenvalue cumulative ratio and detect the knee point. Finally at the system level, to alleviate the additional overheads introduced by metadata, we first co-design a Golomb-Rice coding [28] based bitmap compression scheme that approaches the Shannon entropy lower bound of sparse bitmaps, and further introduce a specialized streaming loading mechanism for more memory-constrained inference.

Extensive experiments on various MoE-LLMs, including OLMoE-1B-7B [29], Qwen1.5-MoE-A2.7B [30], Mixtral-8$\times$7B [8] and Qwen3.5-35B-A3B [31], demonstrate that our **TriS-MoE** achieves an average 8.43% improvement across various evaluation tasks over the strongest extremely low-bit baseline for dense LLMs. More importantly, under real-quantized inference, **TriS-MoE** reduces the memory footprint by $6.32\times$ on average, which compresses the memory usage of Qwen3.5-35B-A3B to only 10.5 GiB and makes it feasible to run on a single consumer-grade GPU.

2 Related Works

2.1 Quantization for MoE-LLMs

Post-training quantization [32, 33] is an efficient and effective approach to alleviating the storage and inference overhead of MoE-LLMs, which has made great progress recently [34, 35]. QMoE [36] extends GPTQ to MoE-LLMs associated with sparse CUDA kernel which achieves 2-bit weight compression. MxMoE [37] divides the MoE layer into finer-group and leverages group-wise mix-precision quantization including W4A4 or W2A16. MoQa [38] evaluates the importance of experts based on the joint distribution on different datasets, where the most salient experts will be allocated with 8/6/4/2-bit. EAQuant [39] introduces a unified smooth factor to address the diverse outliers distribution among experts under W8A8. BiMoE [40] alleviates the redundancy among experts using output-side backbone. MC-MoE [41] also presents a mix-precision pipeline by bit-widths allocating based on access frequency and activation weight. MiLo [42] proposes to use low-rank error compensator to minimize the large performance drop at 3-bit. Although these methods address important aspects of MoE compression, there still lacks a directly comparable framework for real extremely low-bit PTQ on MoE-LLMs, where sub-2-bit effective storage, metadata overhead, and deployable quantized inference are jointly considered.

2.2 Extremely Low-Bit Quantization for LLMs

Extremely low-bit quantization, which enables sub 2-bit compression, has attracted considerable attention, but it often suffers from significant performance degradation. Although several quantization-aware training (QAT) methods [16, 17, 43] have achieved notable success, their immense computational cost limits broader exploration, which also highlights the promise of effective PTQ methods. PB-LLM [19] explore the importance of salient weights and design extra 1-bit unstructured masks to retain them into 8-bit while binarizing the others. BiLLM [20] presents finer-grained masks to divide into 3 groups for binarization using different scaling factors. ARB-LLM [44] further extends this idea by applying fine-grained grouping to salient weights as well to improve performance. However, these dense-LLMs methods overlook the distinctive properties of MoE-LLMs, such as shared input-side sensitive structures and narrower activation subspaces, which results in suboptimal results. Moreover, despite sub-2-bit weights, the metadata introduced for accuracy recovery, including grouping information and the scaling factors, becomes a dominant deployment overhead, pushing the effective memory footprint close to 3 bits. To achieve truly extremely low-bit MoE quantization, we propose **TriS-MoE** to address these limitations. More discussion please refer to Appendix G.

3 TriS-MoE

In this section, we provide a detailed introduction to our **TriS-MoE**. In Section 3.1, we propose a shared high-precision input-side backbone to preserve activation-sensitive input geometry. In Section 3.2, we introduce a routed null space error redirection strategy to alleviate the binarization errors. At last, we describe our metadata compression and real-quantized inference mechanisms in Section 3.3.

3.1 Structure: Shared High-Precision Activation-Sensitive Input-Side Backbone

In studies on extremely low-bit PTQ, a common strategy for preserving performance is to design a sensitivity metric and retain the selected salient weights in higher precision [19, 20, 44]. So to satisfy the extremely low-bit budgets, the central question is where the limited high-precision capacity should be allocated. Prior activation-aware quantization studies have shown that weights associated with salient input activations are particularly sensitive to quantization errors [21, 22, 23]. This suggests that, before aggressively binarizing expert weights, it is crucial to first protect the input-side geometry that directly interacts with activations.

However, existing extremely low-bit quantization methods for dense LLMs usually evaluate such sensitivity in each weight matrix independently. When directly extended to MoE-LLMs, this may result in expert-wise independent binarization and compensation. Such a formulation is suboptimal for MoE layers, because although experts in the same layer are output-specialized, they all consume hidden states from the same upstream representation space. Therefore, the input side of these experts is defined over a common hidden coordinate system. This motivates us to allocate the limited high-precision budget at the layer level, instead of repeatedly and separately protecting the input side of every expert which may also introduce unavoidable overheads.

Based on this observation, we construct a shared high-precision input-side backbone for each MoE layer. Specifically, consider a MoE layer i with N experts, where the weights of the e-th expert is denoted as $\mathrm{W}_i^e \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Since all experts take activations from the same hidden space, we first construct the input-side backbone over this shared input coordinate system. We vertically concatenate all expert weights in the same layer and then perform singular value decomposition (SVD):

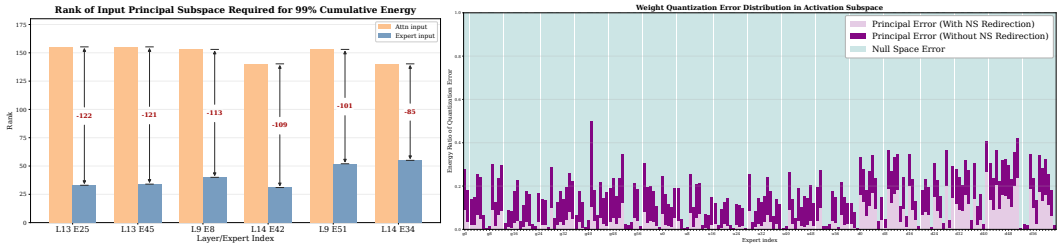
$$\bar{\mathrm{W}}_i = \begin{bmatrix} \mathrm{W}_i^1 \\ \mathrm{W}_i^2 \\ \vdots \\ \mathrm{W}_i^N \end{bmatrix} \in \mathbb{R}^{N d_{\text{out}} \times d_{\text{in}}}, \quad \text{SVD}(\bar{\mathrm{W}}_i) = \begin{bmatrix} \mathrm{U}_i^1 \\ \mathrm{U}_i^2 \\ \vdots \\ \mathrm{U}_i^N \end{bmatrix} (\Sigma_i \mathrm{V}_i^\top), \tag{1}$$

where $\bar{\mathrm{W}}_i$ is the weight matrix of concatenated experts in the i-th layer. Σ_i denotes the singular values. U_i and V_i represents the left and right singular vectors, respectively. By dimensional analysis, we can notice that the factor $\Sigma_i \mathrm{V}_i^\top$ lies on the input side and directly acts on routed activations. Therefore, in order to preserve the sensitive information associated with input activations, we define $\Sigma_i \mathrm{V}_i^\top$ as the **layer-level shared input-side backbone** and save it at 4-bit. Subsequently, the remaining joint left singular vectors are repartitioned into N groups, each corresponding to one of the original experts, which are then binarized to compress the dominant storage and inference overheads. With the shared backbone, we can naturally leverage the unique characteristics of MoE to re-allocate the limited high-precision budget to the activation-sensitive to save performance input geometry without complex design. To further substantiate our design and the importance of input-side preservation, we conduct an additional experiment on Qwen1.5-MoE-A2.7B [30]. As shown in Table 1, our high-precision shared input-side backbone outperforms the shared $\mathrm{U}_i\Sigma_i$ backbone used in [40] which is obtained from horizontally concatenated experts, confirming the advantage of preserving input-side structures.

Table 1: Effect of high-precision backbone choices on Qwen1.5-MoE-A2.7B. Preserving the input-side $\Sigma\mathrm{V}^\top$ backbone gets lower perplexity than the output-side $\mathrm{U}\Sigma$.

Preserved Backbone	WikiText2 ↓	C4 ↓
None (BiLLM)	28.26	95.71
Output-side $\mathrm{U}\Sigma$	20.79	56.11
Input-side $\Sigma\mathrm{V}^\top$	**15.06**	**37.85**

Notably, the shared input-side backbone is not assumed to recover an identical singular structure shared by all experts. Instead, it serves as a layer-level input coordinate system that preserves activation-sensitive input geometry, so that the remaining components can then be pushed to binary form for aggressive compression. While building an expert-specific input-sensitive input-side backbone may perform better, its additional overhead is prohibitive. By contrast, our shared backbone avoids redundant input-side distortions across experts and achieves a more favorable accuracy–compression tradeoff under extremely low-bit budgets.



(a) Rank of Input Principal Subspace

(b) Quantization Error Distribution before and after NS Redirection

Figure 2: Motivation and the effect of our subspace-level design. (a) Experts exhibit much narrower principal subspace; (b) with our design, the weight quantization error interacted with the principal directions of the input activations is redirected into the null space.

3.2 Subspace: Redirecting Expert Quantization Error into Null Space

After preserving the activation-sensitive input-side backbone, the remaining quantization error mainly comes from the binarized expert-specific components. Existing weight-only PTQ methods typically optimize the output reconstruction objective:

$$\arg\min_{\mathrm{W}_q} \|\mathrm{W}\mathrm{X} - \mathrm{W}_q\mathrm{X}\|_2^2, \tag{2}$$

where the weight quantization error is $(\mathrm{W} - \mathrm{W}_q)$. Under extremely low-bit quantization, such numerical error is unavoidable and also difficult to eliminate due to the limited representation capacity of binarized weights. After revisiting Eq. 2, we can observe that the output perturbation is determined not only by the magnitude of the weight error, but also by how this error interacts with the activation subspace. Therefore, the key question now shifts to how to prevent the weight error from affecting the actual expert outputs.

A recent work Q2N [26] provides a fresh perspective on quantization objective that as long as the weight error lies in the null space of the input activations, the quantization error on output will be effectively alleviated, *i.e.,* $\|(\mathrm{W} - \mathrm{W}_q)\mathrm{X}\|_2^2 \approx 0$. To further explore whether this lemma is especially suitable for MoE, we visualize the effective subspaces of the input activations for experts and the self-attention modules in MoE-LLMs. As shown in Figure 2a, the experts exhibit much narrower principal subspaces. We theoretically analysis this phenomenon: Unlike dense layers that process all tokens, each expert only serves a routed subset of tokens. Let X^e denotes the routed activations received by expert e. Due to routing-induced specialization, X^e typically occupies a much narrower subspace than the full layer activation. This leaves a larger unused input subspace for each expert, which we refer to as a larger null space margin. Such a margin relaxes the constraint of mapping quantization errors into the null space. Therefore, it is overly conservative to suppress the expert quantization error over the entire input space for MoE-LLMs under extremely low-bit. The residual error does not need to be eliminated in all directions, but only needs to avoid the principal subspace actually used by the experts. In this way, the quantization error can still exist numerically, but its influence on the routed expert outputs is largely suppressed.

Following existing methods [45, 46, 26], the expert-wise null space of input activation X^e can be modeled as that of their uncentered covariance matrix $\mathrm{X}^e\mathrm{X}^{e\top}$ to guarantee stability. Subsequently, we apply SVD to it: $\mathrm{U}^e, \Sigma^e, \mathrm{V}^e = \mathrm{SVD}(\mathrm{X}^e\mathrm{X}^{e\top})$. Then, extract the column vectors in U^e corresponding to zero singular values to conduct a submatrix U_n^e. With U_n^e, we can get the null space projection operator $\Delta^e = \mathrm{U}_n^e\mathrm{U}_n^{e\top}$, which satisfies: $\|(\mathrm{W}^e - \mathrm{W}_q^e)\Delta^e\mathrm{X}^e\|_2^2 \approx 0$, *i.e.*, Δ^e can map the weights error into the null space of inputs, please refer to Appendix B. However, routed activation matrices in LLMs rarely contain exact zero eigenvalues, making the boundary between the principal subspace and the null space ambiguous. [26] estimate this boundary using a manually selected threshold to select index, but it depends heavily on the threshold choice and may not generalize well across different models, layers, and experts.

To make null space approximation adaptive, we propose a curvature-driven routed null space search method. The intuition is that the cumulative eigenvalue energy curve naturally reflects the transition from informative principal directions to low-energy null space directions. Let λ_j^e denote the eigenval-

ues of $\mathrm{X}^e\mathrm{X}^{e\top}$ sorted in descending order, we define the eigenvalue cumulative ratio $R_e(k)$ of expert e as:

$$R_e(k) = \frac{\sum_{j=1}^{k} \lambda_j^{e2}}{\sum_{j=1}^{n} \lambda_j^{e2}}, \tag{3}$$

where k and n are the index and total amounts of eigenvalues. This curve is monotonically increasing and gradually saturates. The knee point of this curve indicates where the marginal information gain begins to decrease sharply [47]. Therefore, it provides a natural boundary between the principal routed activation subspace and the approximate null space.

However, since $R_e(k)$ is discrete and can be sensitive to local fluctuations, directly computing finite-difference curvature and knee point may lead to unstable estimates. To address this, we first normalize the discrete cumulative ratio curve and then apply the Kneedle algorithm [27] to detect the knee point k_e on the discrete curve:

$$\hat{R}_e(k) = \frac{R_e(k) - R_e(1)}{R_e(n) - R_e(1) + \epsilon}, \qquad \hat{x}_k = \frac{k-1}{n}, \quad k = 1, \ldots, n, \tag{4}$$

$$k_e^* = \arg\max_{1 \le k \le n} \left(\hat{R}_e(k) - \hat{x}_k \right), \tag{5}$$

where k_e^* indicates the point with the largest deviation between the discrete curve with a uniformly increasing linear reference. After getting k_e^*, we redefine the range of U_n^e accordingly to estimate the null space projection matrix Δ based on $\mathrm{U}_n^e\mathrm{U}_n^{e\top}$. However, storing Δ^e, a large scale projection matrix, will incur additional memory overhead. To address this issue, inspired by [26] we extend its close-form solution for equivalent-vector formulation to our MoE quantization:

$$\alpha_e^* = \underset{\alpha_e}{\operatorname{argmin}}(\|(\mathrm{W}^e - \mathrm{W}_q^e) \times \Delta^e - (\mathrm{W}^e - \alpha_e \mathrm{W}_q^e)\|_2^2 + \theta\|\alpha_e - 1\|_2^2) = \frac{\langle \mathrm{W}_q^e, \mathrm{M}^e \rangle + \theta}{\langle \mathrm{W}_q^e, \mathrm{W}_q^e \rangle + \theta}, \tag{6}$$

where θ is the regularization coefficient to balance optimization and $\mathrm{M}^e = \mathrm{W}^e - (\mathrm{W}^e - \mathrm{W}_q^e) \times \Delta^e$. Considering their equivalent-vector may lose some information due to the limited scale, we also present a pair of more accurate quantized low-rank projection matrices through derivation, which may only introduce extra 0.4-bit but performs even better. Please refer to Appendix B for details. For accuracy-compression tradeoff, in this paper we mainly apply the vector-mode for implementation.

As demonstrated in Figure 2b, with this subspace-level design, our **TriS-MoE** exploits the routing-induced low-dimensional activation subspace of each expert and places the unavoidable binarization error into directions that are rarely observed by the routed tokens. Notably, we only apply the null space optimization on experts due to their narrower principal input subspace. We also discover that this design performs unstable on self-attention module, please refer to Appendix B for details.

3.3 System: Metadata-Centric Compression

With the above design, our **TriS-MoE** achieves nominal extremely low-bit quantization. However, when the weights approach 1-bit, quantization metadata [24, 25], including scaling factors, grouping bitmaps, and indexing information, becomes comparable to or even larger than the compressed weights themselves. This issue has been widely ignored by previous methods. Therefore, we co-design a system framework that jointly includes lightweight binarized packing, grouped expert execution, block-level Golomb-Rice bitmap compression, and streaming bitmap loading.

Binarized Packing and Grouped Expert Execution Before introducing bitmap compression, we first design a compact execution layout for binarized linear projections. Existing binarization methods often store unstructured group-wise scaling factors and bitmaps separately, which causes fragmented memory accesses and inefficient per-expert execution. **TriS-MoE** packs 16-bit scaling factors within the same block into contiguous 64-bit words and decodes them through lightweight bitwise operations. During reconstruction, the sign of the scaling factor is adjusted in-place according to the binary weight sign. Meanwhile, instead of launching separate kernels for different experts, we

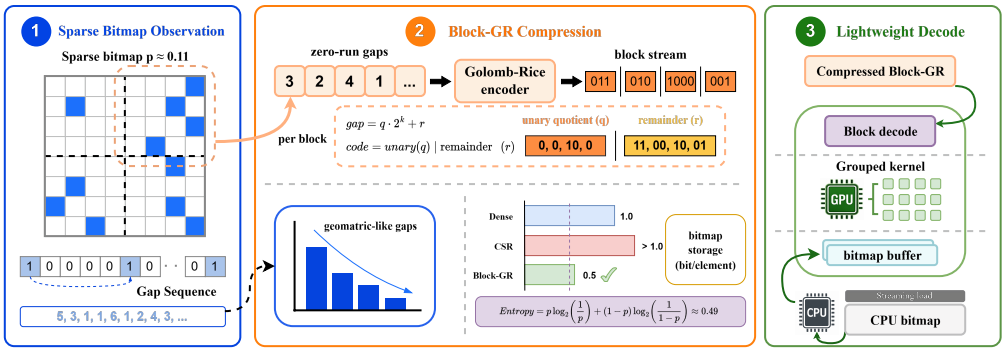


Figure 3: An overview of our real-quantized inference system with metadata-centric compression: 1) the grouping bitmap is highly sparse; 2) we convert the bitmap into zero-run gap sequences and then apply block-level Golomb-Rice coding to compress these gaps; 3) during inference, each kernel tile performs lightweight local decoding, while streaming loading further reduces memory.

reorganize routed tokens into a grouped contiguous layout by expert and cast all activated experts into a segmented batched GEMM-like problem. Inspired by [37], a single CUDA kernel iterates over all expert segments, which improves GPU occupancy and removes fine-grained per-expert scheduling overhead. This compact packing and grouped execution provide the runtime substrate for efficiently consuming compressed metadata, please refer to Appendix D.

Block-GR Bitmap Compression In extremely low-bit quantization, quantization metadata can become the dominant storage bottleneck. Taking binarization methods such as BiLLM as an example, the bitmap used to record unstructured grouping information introduces approximately 1 bit of storage per weight. When the weights approach around 1-bit, this bitmap alone becomes comparable to the weight payload, which prevents the model from real extremely low-bit storage. A natural solution is to exploit bitmap sparsity with CSR encoding. However, we find that CSR does not provide the expected compression in practice. Since each nonzero entry requires additional index metadata, the index overhead can easily exceed the single-bit payload of the original bitmap, and the average storage can even become larger than 1 bit per element, as shown in Figure 3. This motivates us to design a bitmap coding method that matches the actual statistical structure of the grouping bitmap. Empirically, on OLMoE-1B-7B, the proportion of 1's in the bitmap is only $p \approx 0.11$, and the gaps between consecutive 1's approximately follow a geometric distribution. Such a distribution is naturally suited to Golomb-Rice coding [28], as illustrated in Figure 3. Specifically, we convert the bitmap into a gap sequence, where each gap records the number of consecutive zeros between two adjacent 1's. For a gap value g, Golomb-Rice coding with parameter $m = 2^b$ represents the quotient $\lfloor g/m \rfloor$ using unary code and the remainder $g \bmod m$ using b fixed bits. Since most gaps are short under the geometric distribution, they can be represented with compact variable-length codes, significantly reducing the average bitmap storage. The compression performance of Golomb-Rice coding can also be understood from the entropy lower bound. Given the empirical density p, the binary entropy of the bitmap is:

$$H = -p \log_2 p - (1-p) \log_2 (1-p) \approx 0.49 \text{ bit/element}, \tag{7}$$

which is far below the original 1 bit/element dense bitmap.

A remaining challenge is that global variable-length decoding can introduce thread divergence, irregular memory access, and long dependency chains during inference. To make Golomb-Rice coding compatible with GPU kernels, we further propose **block-level Golomb-Rice coding**. The bitmap is partitioned according to inference-kernel tiles, and each block independently collects zero-run gaps and compresses them. During execution, each kernel tile only decodes its corresponding local block, avoiding the need to parse a global code stream. We also adopt a hybrid fallback strategy that sparse blocks are encoded with Golomb-Rice coding, while locally dense blocks fall back to standard bitpacking to avoid worst-case variable-length overhead. With these designs, **TriS-MoE** significantly reduces bitmap metadata storage while preserving stable local decoding and regular memory access during inference.

Table 2: Comparison results on language generation and commonsense reasoning benchmarks. The small numbers in Avg. denote the accuracy drop and perplexity increase compared with the corresponding FP16/BF16 model. Bits only consider weights without metadata.

Model	Method	Weight Bits	Accuracy ↑								PPL ↓		
			A-E	A-C	HS	L-O	L-S	PQ	WG	**Avg.**	Wiki	C4	**Avg.**
OLMoE -1B-7B	FP16	16	76.68	49.74	78.21	70.89	65.52	79.54	68.59	69.88	6.30	11.78	9.04
	GPTQ	2.00	28.28	26.79	26.76	0.00	0.00	51.52	48.30	25.95 43.93↓	24.71	89.59	57.15 48.11↑
	QuIP	2.00	47.98	30.80	51.67	21.81	19.19	64.53	54.85	41.55 28.33↓	15.42	32.73	24.08 15.04↑
	Slim-LLM	2.00	43.18	28.15	50.56	22.91	15.07	64.47	54.14	39.78 30.10↓	16.31	41.14	28.73 19.69↑
	PB-LLM	1.70	37.54	24.06	35.78	31.73	25.85	57.78	52.72	37.92 31.96↓	22.63	53.69	38.16 29.12↑
	BiLLM	1.24	43.18	28.67	45.01	24.45	18.20	62.46	54.85	39.55 30.34↓	18.28	39.46	28.87 19.83↑
	TriS-MoE	**1.33**	**59.18**	**34.64**	**55.95**	**44.71**	**32.66**	**69.64**	**57.85**	**50.66** 19.22↓	**13.76**	**28.35**	**21.05** 12.01↑
Qwen1.5 -MoE-A2.7B	BF16	16	69.19	44.11	77.29	71.40	64.49	80.30	69.30	68.01	6.79	9.74	8.27
	GPTQ	2.00	31.31	23.21	33.28	2.29	1.79	54.24	53.67	28.54 39.47↓	71.29	417.66	244.48 236.21↑
	QuIP	2.00	37.25	25.17	42.42	6.66	6.17	61.43	52.33	33.06 34.95↓	24.02	58.63	41.33 33.06↑
	Slim-LLM	2.00	41.20	25.17	33.59	13.42	3.99	58.43	49.64	32.21 35.81↓	29.79	105.13	67.46 59.19↑
	PB-LLM	1.70	34.30	23.72	32.06	18.82	11.35	57.83	51.14	32.75 35.27↓	91.08	225.00	158.04 149.77↑
	BiLLM	1.24	46.09	28.24	41.78	16.48	14.07	63.55	55.80	38.00 30.01↓	28.26	95.71	61.98 53.71↑
	TriS-MoE	**1.31**	**50.19**	**28.67**	**48.02**	**39.69**	**31.77**	**68.72**	**58.01**	**46.44** 21.57↓	**14.80**	**36.07**	**25.44** 17.17↑
Qwen3.5 -35B-A3B	FP16	16	78.91	62.20	82.42	72.75	69.07	83.19	74.27	74.69	6.26	10.63	8.45
	GPTQ	2.00	48.52	34.12	53.09	16.78	16.13	66.10	53.27	41.14 33.54↓	15.00	37.94	26.47 18.02↑
	QuIP	2.00	68.14	44.97	66.35	44.63	42.23	75.03	61.09	57.49 17.20↓	10.39	19.95	15.17 6.72↑
	Slim-LLM	2.00	58.71	38.56	60.33	49.52	41.27	72.19	61.01	54.51 20.17↓	11.75	25.32	18.54 10.09↑
	PB-LLM	1.70	65.36	37.79	52.64	46.28	44.18	69.47	60.98	53.81 20.87↓	15.55	40.31	27.93 19.48↑
	BiLLM	1.22	67.17	44.62	61.84	58.72	46.85	73.83	64.51	59.65 15.04↓	11.81	24.64	18.23 9.78↑
	TriS-MoE	**1.26**	**72.81**	**47.27**	**67.95**	**65.17**	**50.15**	**76.33**	**66.93**	**63.80** 10.89↓	**9.77**	**18.60**	**14.19** 5.74↑
Mixtral-8×7B	FP16	16	82.79	59.39	84.27	77.06	73.74	83.35	77.27	76.84	3.58	7.02	5.30
	GPTQ	2.00	36.61	24.57	33.86	7.67	7.14	55.55	49.17	30.65 46.19↓	13.01	33.18	23.09 17.79↑
	QuIP	2.00	32.53	22.35	31.70	6.71	5.96	55.11	49.80	29.17 47.67↓	29.87	47.54	38.71 33.41↑
	Slim-LLM	2.00	26.89	24.83	26.22	2.93	0.08	52.29	49.41	26.09 50.75↓	9.40	32.27	20.84 15.54↑
	PB-LLM	1.70	28.66	21.93	27.96	9.66	7.63	51.25	49.57	28.09 48.74↓	42.41	95.54	68.97 63.67↑
	TriS-MoE	**1.77**	**44.28**	**26.02**	**37.97**	**21.50**	**16.49**	**63.66**	**52.18**	**37.44** 39.39↓	**7.14**	**16.82**	**11.98** 6.68↑

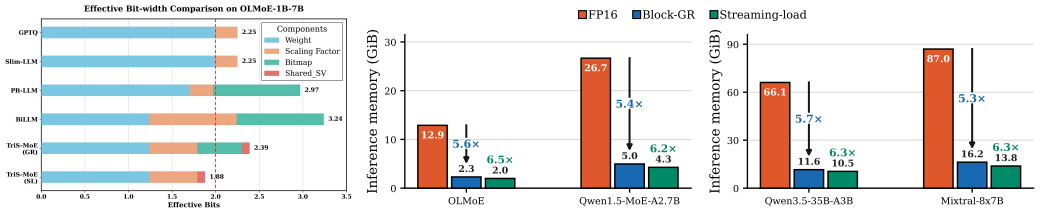
Streaming Bitmap Loading In addition to Block-GR compression, we further support an optional streaming mode for more memory-constrained scenarios, as shown in Figure 3. Specifically, we maintain a two-layer bitmap buffer and asynchronously prefetch the next layer's bitmap through a separate CUDA stream while the current layer is being computed. If the host-to-device transfer is fully overlapped, the bitmap storage on GPU becomes nearly negligible. Unlike Block-GR which provides deterministic compression and stable local decoding, streaming loading trades additional runtime state and scheduling complexity for lower persistent GPU memory. Its benefit may degrade under high CPU load, bandwidth contention, or short decoding-stage compute windows. Therefore, we treat it as an optional aggressive memory-saving mode and discuss its limitations in Appendix D.

4 Experiments

In this section, we conduct extensive experiments to evaluate our **TriS-MoE** on various benchmarks and real systems, which demonstrate our method not only achieves outstanding performance, but also provides a real extremely low-bit pipeline for MoE-LLMs quantization.

4.1 Experimental Setup

Baselines and Models To evaluate our structure and subspace level design for MoE-LLMs, we primarily compare our **TriS-MoE** with PB-LLM [19] and BiLLM [20], two representative sub 2-bit dense-LLMs PTQ methods. In addition, we also include several strong 2-bit baselines, such as GPTQ [12], QuIP [18] and Slim-LLM [48]. We evaluate our method on several widely used MoE models, including OLMoE-1B-7B [29], Mixtral 8×7B [8] and Qwen1.5-MoE-A2.7B [30] and further include the more recent Qwen3.5-35B-A3B [31] to assess its effectiveness on newer architectures.



(a) Effective Bits of Different Methods

(b) Real Inference Memory Reduction of **TriS-MoE**

Figure 4: Real deployment results of **TriS-MoE**. (a) Effective bit-width breakdown on OLMoE-1B-7B, where metadata significantly increases the real bit-width of existing sub 2-bit baselines. **TriS-MoE** reduces this overhead with Block-GR compression or streaming loading. (b) Real inference memory footprint across MoE-LLMs, where **TriS-MoE** achieves clear memory reduction and enables large models such as Qwen3.5-35B-A3B and Mixtral-8×7B to run on a single consumer-grade GPU.

Implementation Details We select BiLLM [20] as our codebase and improve its grouping and scaling granularity following [44]. For our **TriS-MoE** and all baselines, we apply group-wise (128) quantization, and preserve the router in FP16 to prevent routing shifts, while its parameter scale introduces only negligible overheads. Our calibration set consists of 128 random 4096 token-segments from WikiText2 [49] and all quantization procedures are deployed on 1 NVIDIA A100-40GB GPU.

Evaluation Datasets We evaluate language generation and commonsense reasoning capability for the quantized LLMs. For language generation tasks, the test data comes from WikiText2 and C4 [50]. Common reasoning tasks includes ARC [51], HellaSwag [52], LAMBADA [53], MMLU [54], PIQA [55], WinoGrande [56] and MathQA [57], using the open-sourced toolkit lm-evaluation-harness [58]. We also evaluate downstream generation performance on GSM8K [59] and Humaneval [60].

4.2 Comparison Results with Extremely Low-Bit Dense-LLMs Baselines

We first compare **TriS-MoE** with representative low-bit PTQ baselines on language generation tasks. As indicated in Table 2, compared with PB-LLM and BiLLM, which are the most relevant extremely low-bit baselines for dense-LLMs, **TriS-MoE** achieves much lower perplexity on both WikiText2 and C4 across all evaluated MoE models. For example, on Qwen1.5-MoE-A2.7B, **TriS-MoE** reduces WikiText2 perplexity from 28.26 of BiLLM to 14.80, and reduces C4 perplexity from 95.71 to 36.07.

Table 3: Comparison results on MMLU and MathQA.

Method	OLMoE-1B-7B		Qwen3.5-35B-A3B		Mixtral-8×7B	
	MMLU	MathQA	MMLU	MathQA	MMLU	MathQA
GPTQ	23.60	20.10	25.28	23.91	22.17	20.14
Slim-LLM	24.30	22.07	41.02	27.87	22.68	20.84
PB-LLM	22.95	21.41	48.11	29.85	23.06	22.61
BiLLM	23.55	22.35	54.56	32.33	23.01	22.35
TriS-MoE	**27.29**	**23.69**	**61.67**	**37.42**	**23.08**	**22.81**

We further evaluate the reasoning capability on 7 commonsense reasoning tasks. In Table 2, it can be observed that **TriS-MoE** consistently surpasses PB-LLM and BiLLM. Specifically, **TriS-MoE** improves the average reasoning accuracy over BiLLM by 11.12%, 8.44%, 4.15%, and 10.03% on the four models, respectively. Compared with all 2-bit baselines, **TriS-MoE** still achieves an average accuracy improvement of 8.90% across the four models. In addition, Table 3 illustrates the comparison on more downstream tasks such as MMLU and MathQA, and the consistent improvements can be observed. For comparison on GSM8K and HumanEval, please refer to Appendix F. These results indicate that directly applying dense-LLMs PTQ methods to MoE-LLMs is suboptimal, while **TriS-MoE** better preserves the performance through MoE-specific structure and subspace optimization.

4.3 Real-System Results

We further evaluate whether our system-level design can achieve real deployment benefits. As shown in Figure 4a, existing extremely low-bit methods suffer from significant metadata overheads. Although PB-LLM and BiLLM compress the weights to sub 2-bit, their scaling factors and grouping bitmaps significantly increase the effective bit-width, which require over 3.24-bit on OLMoE-1B-7B. In contrast, **TriS-MoE** explicitly compresses the metadata. With Block-GR bitmap compression,

TriS-MoE reduces the effective bit-width to 2.39-bit while preserving stable local decoding. When the optional streaming loading mode is enabled, the persistent bitmap residency on GPU is further removed, which reduces the effective bit-width to 1.88-bit. Please refer to Appendix C for details.

We then measure the real inference memory footprint. As reported in Figure 4b, **TriS-MoE** consistently brings substantial memory reduction. For example, with Block-GR compression, the inference memory is reduced from 66.1 GiB to 11.6 GiB on Qwen3.5-35B-A3B. With streaming loading, the memory further decreases to 10.5 GiB. Overall, **TriS-MoE** achieves average 6.32× memory reduction. More importantly, the two large-scale models, Qwen3.5-35B-A3B and Mixtral-8×7B can be deployed feasibly from A100-class or multi-GPUs to a single consumer-grade GPU. These results verify that **TriS-MoE** not only improves quantization accuracy, but also enables **real** extremely low-bit deployment. Please refer to Appendix D for real-system latency and throughput results.

4.4 Ablation Study

We conduct ablation studies on OLMoE-1B-7B to evaluate the contribution of the structure-level input-side preservation and the subspace-level null space optimization. As shown in Table 4, removing both components leads to severe degradation, with 18.28 and 39.46 perplexity on WikiText2 and C4, respectively. Preserving the input-side backbone substantially reduces the perplexity to 13.91 and 29.30, which demonstrates the importance of activation-sensitive input geometry under extreme-bit quantization. Null space optimization also brings clear improvements by redirecting residual errors away from the routed activation subspace. Finally, combining both components achieves the best results, reaching 13.76 on WikiText2 and 28.35 on C4. These results verify that the structure-level and subspace-level designs are complementary, where the former protects the sensitive input-side backbone while the latter further suppresses the output disturbance caused by unavoidable binarization errors.

Table 4: Ablation studies on OLMoE-1B-7B.

Structure	Subspace	WikiText2 ↓	C4 ↓	**Avg.** ↓
–	–	18.28	39.46	28.87
✓	–	13.91	29.30	21.61
–	✓	14.45	31.17	22.81
✓	✓	**13.76**	**28.35**	**21.05**

5 Conclusion

In this paper, we present **TriS-MoE**, an extremely low-bit PTQ framework for MoE-LLMs. Instead of directly extending dense-model quantization to MoE, **TriS-MoE** revisits the problem from structure, subspace, and system perspectives. It first preserves a shared high-precision input-side backbone to protect activation-sensitive input geometry, then redirects the remaining quantization errors into the routed null space of each expert, and finally compresses quantization metadata with customized system support for real low-bit inference. Extensive experiments on multiple MoE-LLMs demonstrate that **TriS-MoE** consistently outperforms existing extremely low-bit baselines while reducing the real inference memory footprint, which supports Qwen3.5-35B-A3B to run on a single consumer GPU.

Acknowledgement

Miao Zhang was partially sponsored by the National Natural Science Foundation of China under Grant 62676118, 62306084 and U23B2051, Shenzhen Science and Technology Program under Grant KJZD20230923115113026, Guangdong Provincial Key Laboratory of Artificial General Intelligence and Large Models under Grant 2026B1212010005, Guangdong Basic and Applied Basic Research Foundation under Grant 2026A1515011654, and the Fundamental Research Funds for the Central Universities under Grant HITAI4SE2026069. Liqiang Nie was partially sponsored by the Guangdong Science and Technology Program under Grant 2025B0101130003.

References

[1] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

[2] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[3] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

[4] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[7] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[8] 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.

[9] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR, 2023.

[10] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819, 2021.

[11] 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.

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

[13] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pages 7197–7206. PMLR, 2020.

[14] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

[15] Jiaqi Zhao, Chao Zeng, Ming Wang, Linxuan Han, Yuzhang Shang, Miao Zhang, and Liqiang Nie. Lrquant: A unified and learnable framework to post-training quantization for transformer-based large foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[16] Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. Onebit: Towards extremely low-bit large language models. *Advances in Neural Information Processing Systems*, 37:66357–66382, 2024.

[17] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

[18] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429, 2023.

[19] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. Pb-llm: Partially binarized large language models. *arXiv preprint arXiv:2310.00034*, 2023.

[20] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*, 2024.

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

[22] Jiaqi Zhao, Miao Zhang, Ming Wang, Yuzhang Shang, Kaihao Zhang, Weili Guan, Yaowei Wang, and Min Zhang. Ptq1. 61: Push the real limit of extremely low-bit post-training quantization methods for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4483–4502, 2025.

[23] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364, 2024.

[24] Cuong Pham, Anh Dung Hoang, Cuong C Nguyen, Trung Le, Dinh Phung, Gustavo Carneiro, and Thanh-Toan Do. Metaaug: meta-data augmentation for post-training quantization. In *European Conference on Computer Vision*, pages 236–252. Springer, 2024.

[25] Yuzong Chen, Xilai Dai, Chi-chih Chang, Yash Akhauri, and Mohamed S Abdelfattah. The power of negative zero: Datatype customization for quantized large language models. *arXiv preprint arXiv:2501.04052*, 2025.

[26] Jiaqi Zhao, Miao Zhang, Deng Xiang, Ming Li, Weili Guan, and Liqiang Nie. Boost post-training quantization via null space optimization for large language models. *arXiv preprint arXiv:2506.11044*, 2025.

[27] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[28] Solomon Golomb. Run-length encodings (corresp.). *IEEE transactions on information theory*, 12(3):399–401, 1966.

[29] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

[30] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[31] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

[32] Jiaqi Zhao, Miao Zhang, Chao Zeng, Ming Wang, Xuebo Liu, and Liqiang Nie. Lrquant: Learnable and robust post-training quantization for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2240–2255, 2024.

[33] Jiaqi Zhao, Ming Wang, Miao Zhang, Yuzhang Shang, Xuebo Liu, Yaowei Wang, Min Zhang, and Liqiang Nie. Benchmarking post-training quantization in llms: Comprehensive taxonomy, unified evaluation, and comparative analysis. *arXiv preprint arXiv:2502.13178*, 2025.

[34] Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. Mixture of quantized experts (moqe): Complementary effect of low-bit quantization and robustness. *arXiv preprint arXiv:2310.02410*, 2023.

[35] Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12942–12963, 2025.

[36] Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023.

[37] Haojie Duanmu, Xiuhong Li, Zhihang Yuan, Size Zheng, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Mxmoe: Mixed-precision quantization for moe with accuracy and performance co-design. *arXiv preprint arXiv:2505.05799*, 2025.

[38] Zihao Zheng, Xiuping Cui, Size Zheng, Maoliang Li, Jiayu Chen, Xiang Chen, et al. Moqa: Rethinking moe quantization with multi-stage data-model distribution awareness. *arXiv e-prints*, pages arXiv–2503, 2025.

[39] Zhongqian Fu, Tianyi Zhao, Ning Ding, Xianzhi Yu, Xiaosong Li, Yehui Tang, and Yunhe Wang. Eaquant: Enhancing post-training quantization for moe models via expert-aware optimization. *arXiv preprint arXiv:2506.13329*, 2025.

[40] Anonymous. Bimoepushing the limit of post-training quantization for moe-based LLMs, 2025.

[41] 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. *arXiv preprint arXiv:2410.06270*, 2024.

[42] Beichen Huang, Yueming Yuan, Zelei Shao, and Minjia Zhang. Milo: Efficient quantized moe inference with mixture of low-rank compensators. *Proceedings of Machine Learning and Systems*, 7, 2025.

[43] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

[44] Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Linghe Kong, Yulun Zhang, Xiaokang Yang, et al. Arb-llm: Alternating refined binarizations for large language models. *arXiv preprint arXiv:2410.03129*, 2024.

[45] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

[46] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

[47] Jiaquan Wang, Yan Lyu, Chen Li, and Yuheng Jia. Refine and purify: Orthogonal basis optimization with null-space denoising for conditional representation learning. *arXiv preprint arXiv:2602.05464*, 2026.

[48] Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. Slim-llm: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*, 2024.

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

[50] 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.

[51] 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.

[52] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[53] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

[54] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[55] 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, pages 7432–7439, 2020.

[56] 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.

[57] 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, 2019.

[58] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023.

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

[60] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[61] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[62] Carl D Meyer. *Matrix analysis and applied linear algebra*. SIAM, 2023.

[63] Stan Salvador and Philip Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *16th IEEE international conference on tools with artificial intelligence*, pages 576–584. IEEE, 2004.

[64] Peiyu Liu, Zikang Liu, Ze-Feng Gao, Dawei Gao, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Do emergent abilities exist in quantized large language models: An empirical study. *arXiv preprint arXiv:2307.08072*, 2023.

A Preliminaries

Mixture-of-Experts LLMs In MoE-LLMs, the conventional feed-forward-networks are replaced by MoE layers consists of N experts, where the top-k experts are selectively activated for different tokens by the routers. In detail, the output y of each token t in an MoE layer is calculated as:

$$y = \sum_{i \in K} G(t) E_i(t), \tag{8}$$

where K is the top-k routing indices, $E_i(\cdot)$ represents the i-th expert and $G(\cdot)$ denotes the router.

Model Quantization Model quantization aims to convert float weights into corresponding low-bit counterparts to reduce storage and inference overheads. The quantization function can be elaborated as:

$$\mathrm{W}_q = s_q[\text{clamp}(\lfloor \frac{\mathrm{W}}{s_q} \rceil + z_q, 0, 2^b - 1) - z_q], \tag{9}$$

where $\mathrm{W} \in \mathbb{R}^{n \times m}$ and $\mathrm{W}_q \in \mathbb{R}^{n \times m}$ indicate full-precision and fake-quantized weights respectively. $\lfloor \cdot \rceil$ denotes round-to-nearest operator. s_q is the scaling factor and z_q represents zero-point.

Model Binarization Binarization represents the weights are assigned as ± 1 determined by the sign function. In more details:

$$\text{sign}(\mathrm{W}) = \begin{cases} +1, & \mathrm{W} \geq 0, \\ -1, & \mathrm{W} < 0. \end{cases}, \quad \mathrm{W}_b = \alpha \text{sign}(\mathrm{W}), \tag{10}$$

where W_b is fake-binarized weights. α denotes the binarization scaling factors [61, 19, 20] which can be derived analytically by $\alpha_w = \frac{\|w\|_1}{n_w}$, where w is a row of weights with n_w elements.

B Subspace-Level Null Space Error Redirection

B.1 Background: Proofs of Null Space Optimization for LLMs in Prior Works

In this section, to help the readers understand our subspace-level null space error redirection method, we list the detailed proofs in [26] corresponding to the null space projection approximation and the closed-form solution for the equivalent vector.

Proof for the Shared Null Space of X and its Uncentered Covariance Matrix $\mathrm{XX}^\top$

Lemma B.1 *Given input activations $X \in \mathbb{R}^{n \times m}$. Then the null space of X is equal to the null space of its uncentered covariance matrix $XX^\top$, i.e.,*

$$\textit{Null}(X) = \textit{Null}(XX^\top). \tag{11}$$

Proof of Lemma B.1. We show that $w \in \text{Null}(\mathrm{X})$ if and only if $w \in \text{Null}(\mathrm{XX}^\top)$.

(⇒) Suppose $w \in \text{Null}(\mathrm{X})$. Then $w\mathrm{X} = 0$, and thus

$$w\mathrm{X}\mathrm{X}^\top = (w\mathrm{X})\mathrm{X}^\top = 0,$$

which implies $w \in \text{Null}(\mathrm{XX}^\top)$.

(⇐) Conversely, suppose $w \in \text{Null}(\mathrm{XX}^\top)$. Then:

$$0 = w(\mathrm{XX}^\top) = wXX^\top w^\top = (wX)(wX)^\top$$

Since w and X are guaranteed to be nonzero in LLMs, only when $wX = 0$ the equation will hold, *i.e.*, $w \in \text{Null}(X)$.

Therefore, $\text{Null}(\mathrm{X}) = \text{Null}(\mathrm{XX}^\top)$.

Proof for Null Space Projection $\Delta = \mathbf{U}_n\mathbf{U}_n^\top$

Lemma B.2 *$\Delta = U_n U_n^\top$ serves as the null space projection which can project the quantization numerical error of weights into the null space of X, i.e.,*

$$U_n U_n^\top X = \Delta X = (W - W_q)\Delta X = 0.$$

Proof of Lemma B.2. The left singular vector of $\mathrm{X}\mathrm{X}^\top$ is defined as $\mathrm{U} = [\mathrm{U}_p, \mathrm{U}_n]$. Define the singular values $\lambda = \begin{bmatrix} \lambda_p & 0 \\ 0 & \lambda_n \end{bmatrix}$, where all singular values of zero are in λ_n, *i.e.*, $\lambda_n = 0$. Since U is an orthogonal matrix, it can be further derive that:

$$\mathrm{U}_n^\top \mathrm{X}\mathrm{X}^\top = \mathrm{U}_n^\top \mathrm{U}_p \lambda_p \mathrm{U}_p^\top = 0,$$

which indicates that the columns of U_n span the null space for $\mathrm{X}\mathrm{X}^\top$. According to [62], the orthogonal projector of $\mathrm{X}\mathrm{X}^\top$ can be elaborated as:

$$\Delta = \mathrm{U}_n \mathrm{U}_n^\top.$$

Thus $(\mathrm{W} - \mathrm{W}_q)\Delta\mathrm{X} = (\mathrm{W} - \mathrm{W}_q)\mathrm{U}_n\mathrm{U}_n^\top\mathrm{X} = 0$ holds.

Based on Lemma B.1 and Lemma B.2, we can get that $\mathrm{U}_n\mathrm{U}_n^\top$ serves as the null space projection of the input activation.

Theoretical Derivation of the Closed-Form Solution for the Equivalent Projection Vector To satisfy practical inference constraints, the equivalent projection vector α must operate directly on the quantized weights (W_q) and achieve the same effect as the null space projection Δ applied to the post-quantization weight perturbation ($\mathrm{W} - \mathrm{W}_q$), so the objective function is formulated as:

$$\mathcal{L}(\alpha) = \|(\mathrm{W} - \mathrm{W}_q) \times \Delta - (\mathrm{W} - \alpha\mathrm{W}_q)\|_2^2 + \lambda(\alpha - 1)^2\mathrm{I},$$

where the second term is the regularization term avoiding disturbing prior optimizations. Define $\mathrm{M} = \mathrm{W} - (\mathrm{W} - \mathrm{W}_q) \times \Delta$, take the derivative of $\mathcal{L}$ with respect to each dimension i of α and set it to zero:

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = -2\langle \mathrm{W}_q^i, \mathrm{H}^i \rangle + 2\alpha_i \langle \mathrm{W}_q^i, \mathrm{W}_q^i \rangle + 2\lambda(\alpha_i - 1) = 0.$$

Rearranging the terms:

$$\alpha_i(\langle \mathrm{W}_q^i, \mathrm{W}_q^i \rangle + \lambda) = \langle \mathrm{W}_q^i, \mathrm{M}^i \rangle + \lambda.$$

Solving for α_i, the closed-form solution for the optimal equivalent projection vector α^* is:

$$\alpha_i^* = \frac{\langle \mathrm{W}_q^i, \mathrm{M}^i \rangle + \lambda}{\langle \mathrm{W}_q^i, \mathrm{W}_q^i \rangle + \lambda}.$$

Applying α^* to W_q, the null space optimization for quantized model can be practically meaningful.

B.2 Low-rank-Matrix Mode Null Space Redirection

In the main paper, we implement null space error redirection with an equivalent-vector formulation from [26] to avoid introducing additional matrix parameters. Although this vector-mode is lightweight and deployment-friendly, it compresses the effect of the null space projection into a single vector, which may lose part of the projection information. In this section, we provide a more accurate low-rank-matrix mode, which explicitly preserves the projection effect with two low-rank matrices while still avoiding the storage of a full projection matrix.

Recall that for an expert weight matrix W^e and its quantized counterpart W_q^e, the quantization error is defined as:

$$\mathrm{E}^e = \mathrm{W}^e - \mathrm{W}_q^e. \tag{12}$$

Let Δ^e denote the estimated null space projection matrix of the routed expert activation. Ideally, if we can construct a compensated quantized weight $\widetilde{\mathrm{W}}_q^e$ such that:

$$\mathrm{W}^e - \widetilde{\mathrm{W}}_q^e = \mathrm{E}^e\Delta^e, \tag{13}$$

Table 5: Comparison between the vector-mode and low-rank matrix-mode null space redirection on four MoE-LLMs. The vector-mode is the default design used in the main paper, while the matrix-mode provides a more accurate null space compensation with moderately increased effective bit-width.

Model	Method	Effective Bits	Accuracy ↑								PPL ↓		
			AE	AC	HS	LO	LS	PI	WG	**Avg.**	Wiki	C4	**Avg.**
OLMoE -1B-7B	BiLLM	3.24	43.18	28.67	45.01	24.45	18.20	62.46	54.85	39.55	18.28	39.46	28.87
	TriS-MoE (Vector)	1.88	59.18	34.64	55.95	44.71	32.66	69.64	57.85	50.66	13.76	28.35	21.05
	TriS-MoE (Matrix)	2.24	**60.82**	**34.04**	**56.55**	**49.02**	**33.55**	**70.40**	**58.48**	**51.84**	**12.91**	**25.96**	**19.44**
Qwen1.5 -MoE-A2.7B	BiLLM	3.24	46.09	28.24	41.78	16.48	14.07	63.55	55.80	38.00	28.26	95.71	61.98
	TriS-MoE (Vector)	1.86	50.19	28.67	48.02	39.69	31.77	68.72	58.01	46.44	14.80	36.07	25.44
	TriS-MoE (Matrix)	2.26	**54.92**	**30.72**	**48.31**	**42.09**	**34.89**	**70.13**	**58.72**	**48.54**	**13.59**	**33.48**	**23.54**
Qwen3.5 -35B-A3B	BiLLM	3.22	67.17	44.62	61.84	58.72	46.85	73.83	64.51	59.65	11.81	24.64	18.23
	TriS-MoE (Vector)	1.81	**72.81**	**47.27**	**67.95**	65.17	**50.15**	76.33	66.93	**63.80**	**9.77**	18.60	14.19
	TriS-MoE (Matrix)	2.21	71.59	47.01	68.74	**66.10**	47.80	**77.78**	**67.25**	63.75	9.89	**17.94**	**13.92**
Mixtral-8×7B	BiLLM	3.24	28.74	23.46	27.98	6.13	2.91	52.77	49.88	27.41	75.61	194.35	134.98
	TriS-MoE (Vector)	2.32	44.28	26.02	37.97	21.50	16.49	63.66	52.18	37.44	**7.14**	16.82	11.98
	TriS-MoE (Matrix)	2.72	**52.35**	**30.29**	**44.80**	**24.51**	**22.63**	**65.40**	**53.35**	**41.90**	7.26	**16.66**	**11.96**

then the remaining weight error is redirected into the null space of the routed activations, and its output perturbation can be largely suppressed:

$$(\mathrm{W}^e - \widetilde{\mathrm{W}}_q^e)\mathrm{X}_e = \mathrm{E}^e \Delta^e \mathrm{X}^e \approx 0. \tag{14}$$

A direct construction is:

$$\widetilde{\mathrm{W}}_q^e = \mathrm{W}^e - \mathrm{E}^e \Delta^e. \tag{15}$$

However, explicitly storing $\Delta^e \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{in}}}$ is prohibitively expensive. To derive a compact form, we decompose the activation subspace into the principal subspace and the approximate null space. Let $\mathrm{U}_p^e \in \mathbb{R}^{d_{\mathrm{in}} \times k_e}$ be the principal basis selected by the knee point in Section 3.2, and U_n^e be the corresponding approximate null space basis. Since the singular vectors form an orthogonal basis, we have:

$$\mathrm{U}_p^e \mathrm{U}_p^{e\top} + \mathrm{U}_n^e \mathrm{U}_n^{e\top} = \mathrm{I}. \tag{16}$$

The null space projection can be written as:

$$\Delta^e = \mathrm{U}_n^e \mathrm{U}_n^{e\top} = \mathrm{I} - \mathrm{U}_p^e \mathrm{U}_p^{e\top}. \tag{17}$$

Substituting this into the ideal compensated weight gives:

$$\widetilde{\mathrm{W}}_q^e = \mathrm{W}^e - \mathrm{E}^e \left(\mathrm{I} - \mathrm{U}_p^e \mathrm{U}_p^{e\top}\right) \tag{18}$$

$$= \mathrm{W}^e - \mathrm{E}^e + \mathrm{E}^e \mathrm{U}_p^e \mathrm{U}_p^{e\top} \tag{19}$$

$$= \mathrm{W}_q^e + \mathrm{E}^e \mathrm{U}_p^e \mathrm{U}_p^{e\top}. \tag{20}$$

Therefore, the full compensation term can be represented by two low-rank matrices:

$$\widetilde{\mathrm{W}}_q^e = \mathrm{W}_q^e + \mathrm{A}^e \mathrm{B}^e, \tag{21}$$

where

$$\mathrm{A}^e = \mathrm{E}^e \mathrm{U}_p^e \in \mathbb{R}^{d_{\mathrm{out}} \times k^e}, \qquad \mathrm{B}^e = \mathrm{U}_p^{e\top} \in \mathbb{R}^{k^e \times d_{\mathrm{in}}}. \tag{22}$$

Here k^e is the dimension of the principal routed activation subspace, which is typically much smaller than d_{in} for MoE experts due to routing-induced activation concentration. Thus, $\mathrm{A}^e\mathrm{B}^e$ provides a compact low-rank correction to the quantized weight, without storing the dense projection matrix Δ^e.

During inference, the expert output can be computed as

$$\widetilde{\mathrm{W}}_q^e \mathrm{X}^e = \mathrm{W}_q^e \mathrm{X}^e + \mathrm{A}^e(\mathrm{B}^e \mathrm{X}^e). \tag{23}$$

This form only introduces a low-rank residual branch in addition to the original quantized expert computation. Since $k_e \ll d_{\mathrm{in}}$, the extra computation and storage are limited. In our implementation, both A_e and B_e can be quantized to 4-bit, making the additional bit-width overhead small.

Table 6: Effect of applying null space redirection to self-attention on OLMoE-1B-7B.

Method	WikiText2 ↓	C4 ↓	**Avg.** ↓
Self-Attn + Expert	14.62	29.62	22.12
TriS-MoE (Expert only)	**13.76**	**28.35**	**21.05**

Table 7: Sensitivity to different null space boundary approximation strategies on OLMoE-1B-7B.

Method	**A-E**	**A-C**	**HS**	**WG**	**PQ**	**L-O**	**L-S**	**Avg.**	**Wiki**
Curvature [47]	57.74	32.76	54.49	57.06	67.63	41.12	32.73	49.07	14.46
Q2N [26]	56.69	31.91	52.63	56.62	68.82	41.76	**33.28**	48.82	14.79
Residual [63]	58.50	34.56	53.85	57.70	68.23	41.49	27.71	48.86	14.73
TriS-MoE	**59.18**	**34.64**	**55.95**	**57.85**	**69.64**	**44.71**	32.66	**50.66**	**13.76**

Compared with the equivalent-vector mode used in the main paper, the low-rank-matrix mode preserves the projection structure more faithfully. It directly reconstructs the principal-subspace correction $\mathrm{E}^e\mathrm{U}_p^e\mathrm{U}_p^{e\top}$, so that the final residual error $\mathrm{W}^e - \widetilde{\mathrm{W}}_q^e$ remains closer to the approximated null space. As demonstrated in Table 5, this leads to better performance, but it also introduces additional low-rank parameters and an extra residual computation path. Therefore, we treat it as an accuracy-oriented variant and use the vector-mode as the default implementation in the main paper for a better accuracy–efficiency tradeoff.

B.3 Why Null Space Redirection Performs Unstable on Self-Attention Modules?

We only apply null space redirection to MoE experts because we focus on their unique characteristic whose activation subspace is both narrow and stable, as shown in Figure 2a. To evaluate whether subspace-level optimization is also useful for self-attention module, we conduct a valid experiment. As demonstrated in Table 6, the performance even degrades. While we do not further tune the hyperparameters of the equivalent vectors in [26], this result still indicates the instability of applying them to self-attention modules.

We give a deep analysis in this phenomenon. For MoE experts, each expert only receives the tokens selected by the router. Due to routing-induced specialization, these routed activations are usually concentrated around a more specific semantic or functional region. As a result, the principal activation subspace of each expert is relatively low-dimensional, which leaves a larger and more stable null space margin. This makes the approximated null space more reliable across calibration and inference samples.

In contrast, self-attention blocks process all tokens and are responsible for modeling general contextual interactions. Their inputs cover a much broader representation space, and the corresponding principal subspace is less concentrated, as illustrated in Figure 2a. Therefore, the remaining approximate null space is smaller and more sensitive to the sampled calibration data, which also answers why the performance of [26] is sensitive to hyper-parameter selection. Moreover, self-attention is not merely a set of independent linear transformations. Quantization errors in the query and key projections can change the attention scores and the subsequent softmax distribution, which further affects how information is aggregated across tokens. Therefore, redirecting weight errors into an approximated null space is less reliable for self-attention than for experts.

B.4 Sensitivity Analysis

We further analyze the sensitivity of the design choices in our subspace-level null space redirection. The estimation of the boundary between the principal activation subspace and the approximate null space is a more sensitive component, since it directly determines the directions into which the quantization error is redirected. Therefore, we compare our Kneedle-based boundary approximation with several alternative strategies on OLMoE-1B-7B. As shown in Table 7, our method achieves the highest average accuracy and the lowest WikiText2 perplexity among all compared strategies. This result supports the use of an adaptive expert-wise boundary rather than directly adopting a generic boundary estimation rule. We also study the sensitivity to the regularization coefficient θ in the equivalent-vector compensation. Following the setting used in prior null space quantization,

Table 8: Sensitivity to the regularization coefficient θ on OLMoE-1B-7B.

θ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Avg. Acc.	48.84	**50.66**	48.61	49.13	46.83	48.25	48.63	49.06	48.81

Table 9: Robustness to different calibration distributions. We quantize **TriS-MoE** using WikiText2, WinoGrande, and RACE as calibration sets and evaluate the peformance across downstream tasks.

Model	Calibration	A-E	A-C	HS	PQ	L-O	L-S	WG	RACE	Avg.
OLMoE-1B-7B	**WikiText2**	**59.18**	**34.64**	**55.95**	**69.64**	44.71	**32.66**	57.85	32.54	**48.40**
	WinoGrande	50.97	30.63	52.07	68.55	30.06	26.20	**61.33**	29.19	43.63
	RACE	55.30	30.72	54.90	68.55	**46.98**	28.12	57.85	**33.78**	47.03
Qwen1.5-MoE-A2.7B	**WikiText2**	**50.19**	28.67	**48.02**	68.72	**39.69**	**31.77**	58.01	31.58	**44.58**
	WinoGrande	49.12	28.16	47.77	**69.94**	34.14	23.56	**61.80**	28.52	42.88
	RACE	49.73	**29.27**	46.54	69.78	39.30	25.60	60.14	**31.67**	44.00

we use $\theta = 0.2$ by default without model-specific tuning. Table 8 shows that the method remains effective over a broad range of θ, while $\theta = 0.2$ provides the best average accuracy. This indicates that **TriS-MoE** does not rely on extensive tuning of this coefficient.

B.5 Robustness to Calibration Distribution Shift

Since the routed null space approximation is estimated from calibration activations, its robustness may depend on the calibration distribution. To validate this issue, we evaluate **TriS-MoE** using different calibration datasets to investigate its behavior under calibration distribution shifts. Specifically, in addition to the default WikiText2 calibration set, we use WinoGrande and RACE as task-specific calibration sets and evaluate the quantized models across a diverse set of downstream tasks. As shown in Table 9, task-specific calibration generally improves performance on the corresponding matched task, but tends to sacrifice performance on other tasks. For example, using WinoGrande for calibration improves WinoGrande accuracy from 57.85% to 61.33% on OLMoE-1B-7B and from 58.01% to 61.80% on Qwen1.5-MoE-A2.7B. Similarly, RACE calibration improves the corresponding RACE performance. However, these gains do not consistently transfer to other tasks. In contrast, the default WikiText2 calibration achieves the best overall average accuracy on both models, which reaches 48.40% on OLMoE-1B-7B and 44.58% on Qwen1.5-MoE-A2.7B. This result is consistent with the fact that our null space projection is estimated from routed expert activations, where a task-specific calibration set may bias the estimated activation subspace to a particular routing and representation distribution. Therefore, we use general-domain WikiText2 calibration by default to obtain better cross-task robustness.

C Effective Average Bit-Width Calculation

C.1 Average Bit-Width Calculation for TriS-MoE

For a model quantized via **TriS-MoE**, the post-quantization memory footprint consists of four components: fixed unquantized weights (e.g., lm_head, embeddings), potentially quantized router weights, the quantized weight matrices themselves, and additional shared $\Sigma\mathrm{V}^\top$ matrices. The average bit-width is computed as follows. Let $\mathcal{W}$ denote the set of quantized weight matrices. The total number of original parameters is $P_{\mathcal{W}} = \sum_{W \in \mathcal{W}} |W|$, and let $B_c(\mathcal{W})$ be the storage bits occupied by the c-th category of quantized representations. In our method, the storage overhead is partitioned into four categories: quantized weights, scaling factors, shared $\Sigma\mathrm{V}^\top$, and bitmap, i.e.,

$$c \in \{\text{weight}, \text{scale}, \Sigma\mathrm{V}^\top, \text{bitmap}\}. \tag{24}$$

The average bit-width of the c-th category is defined as:

$$\bar{b}_c = \frac{B_c(\mathcal{W})}{P_{\mathcal{W}}}, \tag{25}$$

and the overall average bit-width is:

$$\bar{b}_{\text{total}} = \frac{\sum_c B_c(\mathcal{W})}{P_{\mathcal{W}}} = \bar{b}_{\text{weight}} + \bar{b}_{\text{scale}} + \bar{b}_{\text{shared_sv}} + \bar{b}_{\text{bitmap}}. \tag{26}$$

For the weight, scale, and bitmap components, the computation is identical for every quantized weight matrix. Consider a weight matrix $W \in \mathbb{R}^{m\times n}$ with a salient column ratio ρ; then:

$$\bar{b}_{\text{weight}} = \frac{(1+\rho)\cdot mn}{mn} = 1+\rho. \tag{27}$$

Our binarization method performs column-wise block quantization. Let g be the number of blocks. For each block, four distinct sets of 16-bit row and column scaling factors are required: two sets for the dense and sparse groups of non-salient columns, and the other two for the first-order weights and second-order residuals of salient columns. Hence,

$$\bar{b}_{\text{scale}} = \bar{b}_{\text{row}} + \bar{b}_{\text{column}} = 4\cdot 16\frac{m\cdot g+n}{mn} = 64\cdot\left(\frac{g}{n}+\frac{1}{m}\right). \tag{28}$$

Because the shared SV has no original parameters on its own, its average bit-width must be amortized from a per-layer perspective. Let E be the number of experts in a single layer, and let $W_g^e \in \mathbb{R}^{O\times I}$, $W_{\text{U}}^e \in \mathbb{R}^{O\times I}$, $W_d^e \in \mathbb{R}^{I\times O}$ denote the gate, up, and down matrices of the e-th expert, with I the input dimension and O the output dimension of the expert layer. The total original parameter count of the expert component in one layer is:

$$P_{\text{expert}} = \sum_{e=1}^{E}\left(|W_g^e| + |W_u^e| + |W_d^e|\right) = \sum_{e=1}^{E}(OI+OI+IO) = 3EOI. \tag{29}$$

After including other weight matrices (e.g., attention, shared experts) in the denominator, the total original parameter count of quantized matrices in a single layer becomes:

$$P_{\text{layer}} = P_{\text{else}} + P_{\text{expert}} = P_{\text{else}} + 3EOI. \tag{30}$$

Within the same layer, W_g, W_u, and W_d each share a dedicated $\Sigma\text{V}_{\text{shared}}^\top$. Taking W_g as an example, $\Sigma\text{V}_{\text{shared}}^\top \in \mathbb{R}^{r\times I}$, and in our method $r = \min(I, EO)$; thus, under typical conditions $r = I$, meaning that each decomposed expert $U_g^e \in \mathbb{R}^{O\times I}$ retains the same shape as the original weight. The storage overhead of $\Sigma\text{V}_{\text{shared}}^{\top g}$ is then:

$$B_{\text{shared_sv}}^g = b_{\text{sv}}\cdot rI = b_{\text{sv}}\cdot I^2. \tag{31}$$

Amortizing over the entire layer, the overall average bit-width contributed by shared $\Sigma\text{V}^\top$ is:

$$\bar{b}_{\text{shared_sv}} = \frac{B_{\text{shared_sv}}^g + B_{\text{shared_sv}}^u + B_{\text{shared_sv}}^d}{P_{\text{layer}}} = \frac{b_{\text{sv}}\cdot(I^2+I^2+O^2)}{P_{\text{else}}+3EOI}, \tag{32}$$

where b_{sv} denotes the bit-width of the quantized $\Sigma\text{V}_{\text{shared}}^\top$, which is 4 bits in our method. P_{else} needs to be adjusted dynamically according to the attention modules and the number of shared experts in different models. The metadata resulting from quantizing $\Sigma\text{V}_{\text{shared}}^\top$ is of negligible magnitude and can be ignored. Concerning the bitmap storage overhead, in the uncompressed state a bitmap requires on average 1 bit per weight. After applying block-group reduction (block-gr), the bitmap cost can be reduced to approximately 0.5 bits. Under streaming bitmap loading, only a two-chunk bitmap window is resident on GPU. The GPU-resident average bit-width of bitmap metadata becomes virtually negligible for large models with many layers.

C.2 Average Bit-Width Calculation for Other Methods

Slim-LLM shares a common origin with GPTQ, and under the default group size of 128, the average bit-width consumed by its scaling factors and zero points is 0.25, leading to a total average bit-width of 2.25. QuIP employs its core symmetric normalized quantization strategy, which requires only one global scaling factor per weight matrix:

$$\bar{b}_{\text{scale}} = \frac{16}{OI}, \tag{33}$$

so that the average scaling factor bit-width is negligible and the total average bit-width is approximately 2 bits. PB-LLM and BiLLM exhibit similar structural compositions after quantization, consisting of three parts: quantized weights, scaling factors (including three types of metadata: scaling factors, zero points, and means), and bitmaps. For the bitmap part, both methods need to

store one bitmap of average 1 bit per quantized weight matrix. The average bit-widths of quantized weights and scaling factors need to be discussed separately. For PB-LLM, we retain a proportion ρ of high-precision weights in the experiments. According to its formulation, low-precision weights require storing scaling factors and means, while high-precision weights store scaling factors and zero points. Let g be the number of groups, $\bar{b}_{\text{scale}}$ be the total average bit-width of scaling factors and zero points, b_h the bit-width of high-precision weights, and $W \in \mathbb{R}^{O\times I}$. The average bit-widths are:

$$\bar{b}_{\text{weight}} = 1 \cdot (1-\rho) + b_h\rho, \tag{34}$$

$$\bar{b}_{\text{scale}} = 16 \cdot \frac{2O + 2Og}{OI} = 32 \cdot \frac{1+g}{I}. \tag{35}$$

BiLLM relies on 1-bit second-order residuals for salient columns to reduce quantization error and partitions non-salient weights into dense and sparse groups, requiring four distinct sets of scaling factors and means in total. Moreover, BiLLM adopts a blocking strategy to enhance quantization fidelity. Within each group of every block, BiLLM requires storing one scaling factor and one mean per output channel. Let g be the number of blocks, $\bar{b}_{\text{scale}}$ be the total average bit-width of scaling factors and means, and ρ be the salient column ratio. The average bit-widths are:

$$\bar{b}_{\text{weight}} = 1 + \rho, \tag{36}$$

$$\bar{b}_{\text{scale}} = 16 \cdot 2 \cdot \frac{4Og}{OI} = 128 \cdot \frac{g}{I}. \tag{37}$$

D System-Level Implementation Details

D.1 Binarized Packing

TriS-MoE converts binarized linear projections into a compact execution-friendly layout. This packing strategy is used not only for MoE expert projections, but also for binarized non-expert projections such as attention projection layers. For a binarized weight matrix, the binary signs are stored as bit-packed words, while the associated scaling factors are grouped according to the same block structure used during quantization. Instead of keeping multiple scaling tensors as independent memory streams, **TriS-MoE** packs several half-precision scaling factors from the same block into a single 64-bit word. Specifically, for four block-local scaling factors $\alpha_{b,1}, \alpha_{b,2}, \alpha_{b,3}, \alpha_{b,4} \in \mathrm{FP16}$,we store their raw 16-bit representations in one word:

$$\mathcal{P}_b = \mathrm{bits}_{16}(\alpha_{b,1}) \mid \big(\mathrm{bits}_{16}(\alpha_{b,2}) \ll 16\big) \mid \big(\mathrm{bits}_{16}(\alpha_{b,3}) \ll 32\big) \mid \big(\mathrm{bits}_{16}(\alpha_{b,4}) \ll 48\big), \tag{38}$$

where $\mathrm{bits}_{16}(\cdot)$ denotes the unsigned 16-bit representation of an FP16 value. At inference time, the kernel recovers these values using lightweight bit extraction and type reinterpretation, allowing the scaling metadata to be consumed with coalesced memory accesses and low decoding overhead.

The packed layout separates salient and non-salient regions. Columns selected by the salient mask are stored in a compact binary representation, while the non-salient binary mask are compressed as sparse bitmap that indicate which binary component contributes to each weight entry. Prefix metadata records the number of active entries per packed block, allowing the kernel to locate the corresponding binary words without scanning dense masks. This design avoids materializing dense binary mask during inference and keeps the working set proportional to the compressed representation.

TriS-MoE also folds vector-form null-space compensation into the packed scaling factors whenever possible. Specifically, row-side and column-side compensation factors are multiplied into the corresponding row and column scales during packing. During reconstruction inside the kernel, the sign of each recovered scale is adjusted according to the binary weight sign in-place, which avoids extra memory traffic for explicitly reconstructed signed weights.

D.2 Grouped Expert Execution

A standard MoE implementation often executes experts independently after routing. This creates many small matrix multiplication problems, especially during autoregressive decoding, where each expert may receive only a few tokens. Launching separate kernels for individual experts introduces scheduling overhead and leads to poor GPU utilization.

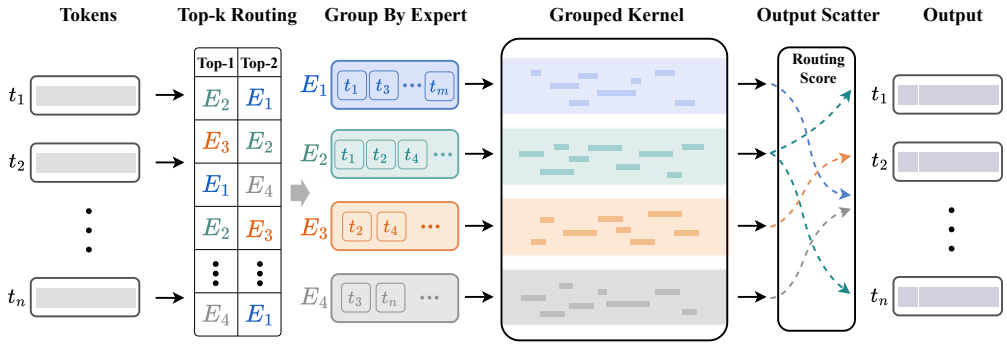


Figure 5: An overview of grouped expert execution in **TriS-MoE**.

As illustrated in Figure 5, **TriS-MoE** avoids this by reorganizing all routed token–expert assignments into a single grouped layout. After the router selects the top-k experts for each token, **TriS-MoE** flattens the token–expert pairs and sorts them by expert id. This produces a contiguous token buffer where tokens assigned to the same expert form one segment. For each active expert, **TriS-MoE** records the segment boundary and splits the segment into fixed-size row tiles. The resulting metadata describes a segmented batched GEMM-like problem: each segment uses a different expert weight, while all active segments are processed by the same grouped kernel launch.

The grouped kernel maps one grid dimension to output-channel tiles and another grid dimension to expert-segment tiles. Inside the kernel, a lightweight lookup maps each problem tile to its corresponding expert segment. The kernel then loads the packed binary payload for that expert, decodes the block-local scaling factors, reconstructs the signed contribution on the fly, and accumulates the result for the routed tokens in that segment. Since all active experts share the same launch, the runtime avoids fine-grained per-expert scheduling and improves parallelism for small or imbalanced routing patterns.

The binarized packing format and grouped expert execution address complementary bottlenecks. The packing format provides a compact runtime layout for binarized linear projections, while grouped execution reduces the scheduling overhead caused by fragmented expert workloads. Together, they provide the runtime substrate for efficient low-bit MoE inference. We report detailed end-to-end latency and throughput evaluations in the next subsection.

D.3 End-to-End Inference Latency and Throughput

To further evaluate whether the proposed system-level design translates the memory benefits of extremely low-bit quantization into practical runtime improvements, we measure the complete end-to-end generation latency and throughput of TriS-MoE. All experiments are conducted on OLMoE-1B-7B using a single NVIDIA A100-40GB GPU under different input lengths, output lengths and batch sizes. In addition to the FP16 model, we include AutoGPTQ-INT4 as a standard low-bit inference baseline under the same hardware and inference settings.

As shown in Table 10, **TriS-MoE** consistently reduces end-to-end generation latency and improves throughput across all evaluated settings. Compared with FP16 inference, **TriS-MoE** achieves $1.30\times$–$2.41\times$ speedup. More importantly, the improvement is not solely due to lower bit-width. Compared with AutoGPTQ-INT4 under the same settings, **TriS-MoE** still achieves $2.15\times$–$3.00\times$ speedup. The improvement remains consistent across different sequence lengths and batch sizes, demonstrating that the proposed low-bit representation can be effectively translated into practical end-to-end inference benefits. The runtime improvement mainly comes from the system-level design. In particular, the grouped expert execution reorganizes fragmented routed expert computations into a contiguous segmented workload, while the compact low-bit packing reduces weight and metadata traffic during inference. These results complement the memory measurements in Section 4.3 and demonstrate that **TriS-MoE** provides practical benefits in both memory footprint and end-to-end inference efficiency.

Table 10: End-to-end generation latency and throughput on OLMoE-1B-7B using a single NVIDIA A100-40GB GPU. All methods are evaluated under the same input/output sequence lengths and batch sizes. Speedup is measured relative to FP16 and AutoGPTQ-INT4, respectively.

Input Length	Output Length	Batch Size	Latency (s) ↓			Throughput (tok/s) ↑			Speedup	
			FP16	INT4	**TriS-MoE**	FP16	INT4	**TriS-MoE**	vs. FP16	vs. INT4
128	512	1	90.79	113.10	**37.70**	5.64	4.54	**13.58**	**2.41**×	**3.00**×
128	2048	2	351.89	407.29	**189.44**	11.64	10.07	**21.62**	**1.86**×	**2.15**×
128	2048	4	397.14	488.52	**222.94**	20.63	17.08	**36.75**	**1.78**×	**2.19**×
2048	512	1	68.17	112.06	**38.92**	7.51	4.59	**13.16**	**1.75**×	**2.88**×
2048	512	2	65.64	109.95	**50.41**	15.60	9.36	**20.31**	**1.30**×	**2.18**×
2048	2048	2	274.12	453.17	**191.88**	14.94	9.04	**21.35**	**1.43**×	**2.36**×

Table 11: Calibration overhead of **TriS-MoE** on a single NVIDIA A100-40GB GPU. We report the runtime of structure-level backbone extraction and subspace-level redirection. In addition, the overall calibration runtime and peak GPU memory of the complete quantization pipeline are also reported.

Model	Component	Per-layer/expert Time (s)	Total Time (s)
OLMoE-1B-7B	Structure	0.47	22.50
	Subspace	0.38	117.56
Qwen3.5-35B-A3B	Structure	0.68	79.82
	Subspace	0.39	1179.43

Model	Calibration Runtime (h)	Peak Memory (GiB)
OLMoE-1B-7B	3.42	12.20
Qwen1.5-MoE-A2.7B	5.80	11.79
Qwen3.5-35B-A3B	13.68	35.83
Mixtral-8×7B	3.21	33.71

D.4 Limitations of streaming bitmap loading

When enabled, streaming bitmap loading reduces GPU memory residency by keeping selected bitmap metadata in host memory and transferring it to GPU memory on demand during inference. However, its benefit depends on whether the host-to-device transfer for the next layer can be hidden behind the computation of the current layer.

Under CPU contention, saturated transfer bandwidth, or copy-engine interference, the prefetch latency may become exposed, leading to unstable throughput. This effect is especially pronounced during autoregressive decoding, where each layer has a short computation window, if the next layer's bitmap has not arrived on GPU when needed, execution must wait for the pending transfer, increasing end-to-end latency. Moreover, streaming bitmap loading requires maintaining additional runtime state, including resident-window management, asynchronous prefetch, event synchronization, and eviction. These mechanisms increase implementation complexity and make performance more dependent on hardware bandwidth and serving-time load. Therefore, streaming bitmap loading should be viewed as a memory-saving deployment option rather than a universally faster inference path.

E Calibration Overhead

We further evaluate their calibration cost introduced by our structure and subspace level design to quantify the practical overhead of TriS-MoE. Table 11 reports the calibration runtime and peak GPU memory consumption. For the two methods, we separately measure their per-layer and total runtime on OLMoE-1B-7B and Qwen3.5-35B-A3B. The SVD-based backbone extraction requires only 22.50 s and 79.82 s in total on the two models, respectively, while the null space redirection takes 117.56 s and 1179.43 s. Compared with the overall quantization procedure, these additional operations constitute only a small fraction of the total calibration time. We further report the complete calibration cost across all four evaluated MoE-LLMs. **TriS-MoE** finishes the entire post-training quantization procedure in 3.21–13.68 hours on a single NVIDIA A100-40GB GPU. The peak memory

consumption ranges from 11.79 GiB to 35.83 GiB. In particular, both Qwen3.5-35B-A3B and Mixtral-8×7B can be fully quantized in the memory capacity of a single A100-40GB GPU. These results show that our structure and subspace level operations remain practical for offline PTQ and do not require multi-GPU calibration.

F More Evaluations on GSM8K and HumanEval

In addition to the benchmarks reported in the main paper, we further evaluate **TriS-MoE** on GSM8K and HumanEval, which require more complex mathematical reasoning and code generation abilities. As shown in Table 12, under extremely low-bit post-training quantization, all PTQ methods suffer severe degradation on relatively small MoE models, and their performance on these two tasks is often close to random or nearly collapsed. This phenomenon is consistent with prior observations that emergent and generation-sensitive abilities are much more fragile under low-bit quantization than standard language modeling or commonsense reasoning tasks [64, 22]. GSM8K and HumanEval are qualitatively different from standard zero-shot classification tasks, where they rely heavily on multi-step reasoning, precise intermediate states, code syntax, and instruction-following behavior. Small perturbations introduced by extremely low-bit PTQ can be amplified throughout the generation process, which leading to incorrect final answers even when the model still preserves reasonable token-level prediction or commonsense recognition ability. Recovering such complex downstream abilities under extreme compression usually requires additional adaptation, such as quantization-aware training, task-specific fine-tuning, or distillation, which is beyond the training-free PTQ setting considered in this work. Nevertheless, on the larger Qwen3.5-35B-A3B model, **TriS-MoE** achieves clearly better performance than existing PTQ baselines, indicating that the preservation of these challenging abilities is strongly related to the capability margin of the original full-precision model. Larger MoE-LLMs provide more redundancy and stronger pre-quantization reasoning/code ability, making them more robust to extreme compression. Therefore, we report GSM8K and HumanEval results in the appendix as additional stress-test evaluations, while keeping the main paper focused on language modeling, commonsense reasoning, real effective bit-width, and deployment-oriented memory results.

Table 12: Evaluations on GSM8K and HumanEval.

Models	Method	GSM8K	HumanEval
OLMoE-1B-7B	Slim-LLM	0.98	0
	BiLLM	1.59	0
	TriS-MoE	**1.67**	0
Qwen3.5-35B-A3B	Slim-LLM	1.89	1.83
	BiLLM	5.30	5.48
	TriS-MoE	**15.47**	**10.37**
Qwen1.5-MoE-A2.7B	Slim-LLM	1.21	0
	BiLLM	0.22	0
	TriS-MoE	**2.05**	0

G Discussion on the Scope and Extensibility of TriS-MoE

We further clarify why the main comparison is conducted against representative dense extremely low-bit PTQ methods, rather than directly integrating every existing MoE-specific compression method. **TriS-MoE** is designed to study the mismatches between current dense-LLMs extremely low-bit baselines with the unique characteristics of MoE-LLMs, and how it should be reformulated. Therefore, we select a BiLLM-style binarization framework as the base quantization pipeline and introduce three MoE-specific improvements on top of it, including shared high-precision input-side backbone, routed null space error redirection, and system-level metadata compression. This design choice allows us to isolate the contribution of each proposed component under the same extreme-bit quantization setting. Since our goal is to validate the effectiveness of the proposed MoE-specific reformulation, we keep the main comparison focused on directly applicable low-bit PTQ baselines and discuss the relationship to MoE-specific compression methods separately.

Table 13 further summarizes the differences between **TriS-MoE** and representative quantization methods. Existing dense extremely low-bit PTQ methods such as PB-LLM and BiLLM support full-model aggressive quantization, but do not explicitly exploit MoE-specific structural properties or optimize the metadata overhead arising at extremely low bit widths. Existing MoE-specific methods, on the other hand, mainly focus on expert-aware mixed-precision quantization, routing-aware optimization, or efficient expert execution, and typically do not jointly consider full-model extremely low-bit PTQ, metadata-aware optimization, and dedicated inference support. In contrast, **TriS-MoE** jointly addresses these aspects from the structure, subspace, and system perspectives.

Table 13: Comparison with representative extremely low-bit and MoE quantization methods.

Method	MoE-Specific Design	Extremely Low-Bit PTQ	Full-Model Quantization	Metadata-Aware Optimization	Dedicated Kernel
PB-LLM [19]	×	✓	✓	×	×
BiLLM [20]	×	✓	✓	×	×
MxMoE [37]	✓	×	×	×	✓
MC-MoE [41]	✓	✓	×	×	✓
EAC-MoE [35]	✓	×	×	×	×
BiMoE [40]	✓	✓	✓	×	×
MiLo [42]	✓	×	✓	×	✓
EAQuant [39]	✓	×	✓	×	×
TriS-MoE	✓	✓	✓	✓	✓

Our subspace-level design is also different from directly applying prior null space PTQ [26] to MoE-LLMs. Prior null space optimization treats the activation subspace as a general property of linear layers, while **TriS-MoE** specifically exploits the narrower and heterogeneous activation subspaces introduced by sparse expert routing. We estimate the null space boundary independently for each routed expert using the proposed Kneedle-based strategy and apply error redirection only to expert modules. As shown in Appendix B.3, extending the same strategy to self-attention modules instead degrades performance, indicating that its effectiveness is closely related to the routing-induced expert subspace rather than null space optimization alone. Furthermore, the sensitivity analysis in Appendix B.4 shows that our adaptive boundary approximation outperforms alternative boundary approximation strategies.

Importantly, our proposed components are not tied to our codebase BiLLM itself. The input-side backbone preservation can be applied to other expert quantization backbones to protect activation-sensitive directions, the routed null space redirection can further compensate residual quantization errors after different quantizers, and the metadata compression scheme can reduce the real storage overhead of methods that rely on fine-grained grouping or salient-weight indicators. Therefore, **TriS-MoE** can also be viewed as a MoE-specific reformulation and system extension for extremely low-bit PTQ, rather than a method that is only applicable to one specific dense baseline. We leave the integration of **TriS-MoE** with more MoE-specific mixed-precision allocation or router-calibration frameworks as a promising future direction.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This paper's scope is clearly reflected in the abstract and the introduction reflects the paper's contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have described the limitation in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide proofs in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the experimental details in Section 3.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the entire codes in the supplementary materials

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the experimental settings in Section 3.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the statistical significance of the experiments following a unified evaluation framework in Section 3.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computing resources cost in Section 3.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and preserved anonymity consistently.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The positive impact is provided in Section 3.2-3.4 and there is no negative impact because model quantization does not amplify hidden biases encoded in LLMs.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There are no such risks in this paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The data, codes and models used in our paper are publicly available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our codes and details are described in the anonymous zip file.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core innovation in this paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.