CLASP: CROSS-LAYER ACTIVE-SUBSPACE POOLING FOR QUANTIZED MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank corrections can recover accuracy lost to aggressive post-training quantization of mixture-of-experts (MoE) language models. Most low-bit MoE correction methods fit their factors independently or within individual layers. We find substantial overlap between experts' activation-weighted dominant subspaces across layers. We use this overlap in Cross-Layer Active-Subspace Pooling (CLASP), which clusters experts by the Grassmannian distance between their active subspaces and fits a shared low-rank basis to each cluster. Sharing reduces basis storage and allows a larger correction rank at a fixed bit budget. In the evaluated Qwen models, experts activated by the same token also tend to belong to a few clusters. CLASP stores the shared factors in a device-resident pool and batches correction operations, with stream overlap reducing execution overhead. At 2.15–2.21 bits per quantized parameter, CLASP achieves the lowest perplexity among the compared quantizers on four MoE models. Under the same measurement protocol and hardware, end-to-end generation is up to $4.9\times$ faster than the FP16 baseline. Our source code will be available after acceptance.

1 INTRODUCTION

Mixture-of-experts (MoE) architectures scale large language models by routing each token to a small subset of expert networks. This increases parameter count without a proportional increase in per-token computation (Shazeer et al., 2017; Fedus et al., 2022; Du et al., 2022; Jiang et al., 2024; Dai et al., 2024). Sparse computation does not remove the need to store expert weights. Even when a token activates only a few experts, all weights must remain resident (Rajbhandari et al., 2022) or be transferred during inference. Memory capacity and bandwidth can therefore limit deployment (Eliseev & Mazur, 2023; Cao et al., 2025).

Post-training quantization (PTQ) reduces model storage without full retraining (Xiao et al., 2025; 2026). Methods developed for dense models include error-compensated rounding (Frantar et al., 2022), activation-aware scaling (Xiao et al., 2023; Lin et al., 2023; Shao et al., 2024), rotations (Ashkboos et al., 2024; Liu et al., 2025), and vector quantization below three bits (Egiazarian et al., 2024; Tseng et al., 2024; Liu et al., 2024). MoE-specific methods adapt these approaches to the distribution of weights and activations across experts (Kim et al., 2023; Hu et al., 2025; Fu et al., 2025). At two bits, quantization alone can still cause substantial accuracy loss.

Recent methods address this loss by combining a quantized backbone with a low-rank correction (Yao et al., 2023; Huang et al., 2025; Gu et al., 2026a;b). The correction represents a small set of directions at higher precision, allowing the remaining weights to be stored at lower bit widths. Several such methods fit the factors without finetuning, keeping the cost of quantization practical.

The storage cost of these corrections depends on how their factors are shared. Existing low-bit MoE approaches use private factors for each expert or share factors among experts within a layer (Gu et al., 2026a). With $L \cdot N$ expert matrices per weight type, repeated bases consume budget that could otherwise support a larger correction rank. Corrections also add work during decoding: small operations for each active expert can incur substantial launch overhead (Huang et al., 2025). We investigate whether similarity-based cross-layer sharing can reduce repeated basis storage while supporting efficient correction execution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

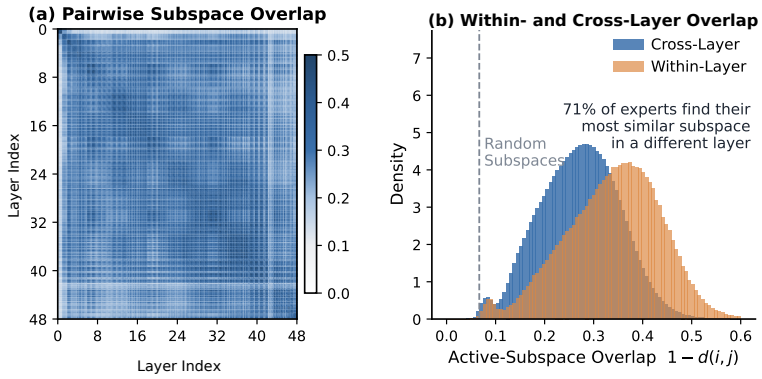

Figure 1: Pairwise overlap of rank-32 active subspaces before clustering, measured on the 6,134 gate-projection expert matrices with calibration statistics in Qwen3-30B-A3B (48×128 experts in total). **(a)** Expert-by-expert overlap matrix with experts ordered by layer; the axes are shown in layer units. High-overlap structure appears across layers. **(b)** Within-layer pairs are more concentrated at high overlap, while the cross-layer distribution remains well above the random-subspace reference. For 71% of experts, the nearest active subspace among all candidates belongs to another layer.

Two observations motivate the design. The activation-weighted dominant subspaces of experts overlap across Transformer layers (Figure 1), suggesting that some of the bases fitted separately within each layer could be shared. In the Qwen models we study, experts activated by a token also concentrate in a few subspace clusters. This grouping gives a small set of shared factors to access during decoding.

CLASP groups experts across layers by the principal-angle distance between their active subspaces. It then fits an activation-weighted low-rank decomposition to each group, storing one shared basis and a private factor per expert. We make two contributions:

- **Cross-layer active-subspace pooling.** Sharing a basis among experts with similar active subspaces reduces correction storage and permits a larger per-expert rank at a fixed bit budget.
- **Batched correction execution.** A device-resident factor pool and batched operations reduce the cost of accessing and applying corrections, with stream overlap further reduce decoding overhead.

On four MoE models, CLASP achieves the lowest perplexity among the compared quantizers at 2.15–2.21 bits per quantized parameter. Its zero-shot accuracy gains vary by model, and end-to-end generation is up to $4.9\times$ faster than the FP16 baseline.

2 Related Work

MoE quantization. MoQE (Kim et al., 2023) studies the tolerance of expert weights to sub-4-bit quantization, and QuantMoE-Bench (Li et al., 2024b) evaluates the sensitivity of different MoE components. MoEQuant (Hu et al., 2025) uses expert-balanced sampling and affinity guidance; EAQuant (Fu et al., 2025) addresses routing consistency and activation outliers. MxMoE (Duanmu et al., 2025) and MC-MoE (Huang et al., 2024) assign mixed bit widths to experts. Other approaches reduce storage or computation by pruning and skipping experts (Lu et al., 2024), merging them using routing information (Li et al., 2023), or offloading them at mixed precision (Tang et al., 2024). MoE-SVD (Li et al., 2025) shares an SVD factor across experts and selects compact expert-specific factors, while TD-MoE (Xu et al., 2026a) jointly factorizes the experts within each layer using a data-aware tensor decomposition. These methods demonstrate the value of shared low-rank structure for MoE compression. Among low-bit quantizers, MiLo (Huang et al., 2025) adds per-expert low-rank corrections, TileQ (Gu et al., 2026a) shares tiled low-rank factors within a layer, and KBVQ-MoE (Xu et al., 2026b) extracts a KLT-guided shared component before vector quanti-

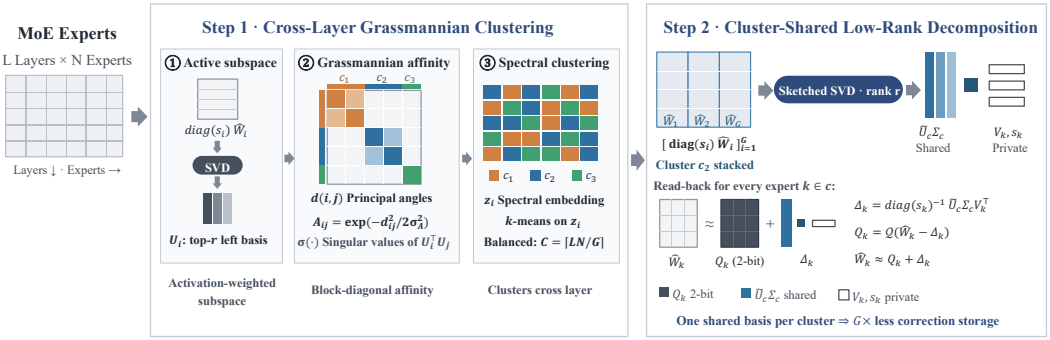

Figure 2: CLASP overview. **Step 1:** extract an active subspace for each expert (①), compute Grassmannian affinities (②), and group experts across layers by spectral clustering followed by a size-balancing step (③). **Step 2:** jointly decompose each cluster into shared factors $(\tilde{U}_c, \Sigma_c)$ and private factors V_k with activation scales s_k. The resulting correction Δ_k is combined with the two-bit backbone Q_k.

zation. CLASP instead forms cross-layer groups from activation-weighted subspace similarity and pools the correction basis within each group.

Low-rank correction of quantization error. Low-rank corrections are related to work on low-dimensional model adaptation (Aghajanyan et al., 2021; Hu et al., 2022) and adapters over quantized backbones (Dettmers et al., 2023). Joint decomposition and initialization methods include LoftQ, LQ-LoRA, and ApiQ (Li et al., 2024c; Guo et al., 2024; Liao et al., 2024). Other methods approximate quantization error directly (Yao et al., 2023; Saha et al., 2024), weight the reconstruction by activations (Zhang et al., 2024; 2025), absorb outliers in diffusion models (Li et al., 2024a), or optimize compensation for two-bit language models (Lee et al., 2025). Cross-layer factor sharing has also been explored in adaptation and compression (Kopiczko et al., 2024; Renduchintala et al., 2024; Wang et al., 2024). GlowQ (An et al., 2026) shares a factor across groups of modules with common inputs in a dense model. CLASP instead groups MoE experts across layers according to active-subspace similarity. We follow LoPRo (Gu et al., 2026b) in fitting the correction before quantizing the residual. SpQR (Dettmers et al., 2024) and LLM.int8() (Dettmers et al., 2022) provide a different approach by retaining selected outlier weights or features at higher precision.

3 Motivation

Layer-local MoE corrections leave open two questions: how much basis storage can be shared across layers, and how cheaply the corrections can be applied during decoding. Earlier cross-layer sharing methods establish that factors can be tied across layers, but do not group MoE experts by the similarity of their active subspaces (Kopiczko et al., 2024; Renduchintala et al., 2024; Wang et al., 2024).

Figure 1 examines the first question using all 6,134 gate-projection expert matrices with calibration statistics in Qwen3-30B-A3B. Same-layer pairs are more concentrated at high overlap, but substantial cross-layer overlap remains throughout the matrix. For 71% of experts, the nearest active subspace among all candidates belongs to another layer. Allowing cross-layer groups therefore gives the pooling procedure access to high-overlap partners that a layer constraint would exclude.

Observations. Active subspaces overlap across layers (Figure 1). In the evaluated Qwen models, a token's routed experts also concentrate in a few of the resulting clusters (Figure 3).

We use the first observation to pool bases across layers (Section 4.2). The second suggests that relatively few shared factors need to be accessed for a token. Section 4.3 describes how the factor pool, batched corrections, and stream scheduling support decoding.

4 METHOD

4.1 PRELIMINARIES AND PROBLEM STATEMENT

Consider an MoE model with L layers of routed experts, N experts per layer, and K weight matrices per expert. We use the row-vector convention $y = x\hat{W}$, with $\hat{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, and approximate each expert weight as

$$\hat{W} \;\approx\; Q + U\Sigma V^{\top}, \tag{1}$$

where $U\Sigma V^{\top}$ is a rank-r correction, $U \in \mathbb{R}^{d_{\text{in}} \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$ is diagonal, $V \in \mathbb{R}^{d_{\text{out}} \times r}$, and Q is the quantized backbone. We fit the correction to the original weights and then quantize the residual, following LoPRo (Gu et al., 2026b). Most of the correction storage lies in U and V. Fitting private factors for all $L \cdot N \cdot K$ expert matrices can therefore be costly, even at a small rank.

4.2 CROSS-LAYER ACTIVE-SUBSPACE POOLING

Cross-layer Grassmannian clustering. Figure 2 summarizes the procedure. We weight the expert matrices by their calibration activations before extracting subspaces. For expert i and input channel q, let $a_{i,q} = \mathbb{E}[|x_q|]^{\alpha}$ and normalize $s_i = a_i / \sqrt{\max_q a_{i,q} \min_q a_{i,q}}$ after clamping values away from zero. Let $U_i \in \mathbb{R}^{d_{\text{in}} \times r}$ be the top-r left singular basis of $\text{diag}(s_i)\hat{W}_i$. We call this the expert's *active subspace*. The normalized Grassmannian principal-angle distance between two such subspaces is (Edelman et al., 1998; Hamm & Lee, 2008)

$$d(i,j) \;=\; \frac{\big\|\arccos \sigma\big(U_i^{\top} U_j\big)\big\|_2}{\sqrt{r} \cdot \pi/2} \;\in\; [0,1], \tag{2}$$

where $\sigma(\cdot)$ returns the singular values. Their inverse cosines are the principal angles between the subspaces (Björck & Golub, 1973). We convert these distances to a Gaussian affinity,

$$A_{ij} \;=\; \exp\!\left(-\frac{d(i,j)^2}{2\sigma_A^2}\right), \tag{3}$$

where σ_A is set to the mean of the positive pairwise distances. We then apply spectral clustering (Ng et al., 2001). With degree matrix $D = \text{diag}(\sum_j A_{ij})$, the normalized Laplacian is $L_{\text{sym}} = I - D^{-1/2} A D^{-1/2}$. Let z_i denote the spectral embedding of expert i obtained from the C eigenvectors with the smallest eigenvalues. We partition the embeddings using k-means,

$$\min_{\{\mathcal{C}_1,\ldots,\mathcal{C}_C\}} \sum_{c=1}^{C} \sum_{i \in \mathcal{C}_c} \big\|z_i - \mu_c\big\|_2^2, \qquad \mu_c = \frac{1}{|\mathcal{C}_c|} \sum_{i \in \mathcal{C}_c} z_i, \tag{4}$$

with $C = \lceil LN/G \rceil$ per weight type. We then order the experts by spectral-cluster assignment and split the ordered list into groups of at most G experts; the final group may be smaller. This size-balancing step is separate from the k-means objective. Experts from any layer may be assigned to the same group.

Cluster-shared low-rank decomposition. For a cluster $\mathcal{C}_c$ with $G_c = |\mathcal{C}_c| \le G$ members, we concatenate its scaled weight matrices and compute a rank-r approximation,

$$\big[\,\text{diag}(s_1)\hat{W}_1 \;\big|\; \cdots \;\big|\; \text{diag}(s_{G_c})\hat{W}_{G_c}\,\big] \;\approx\; \tilde{U}_c\, \Sigma_c\, \tilde{V}_c^{\top}, \tag{5}$$

with $\tilde{U}_c \in \mathbb{R}^{d_{\text{in}} \times r}$, $\Sigma_c \in \mathbb{R}^{r \times r}$, and $\tilde{V}_c \in \mathbb{R}^{G_c d_{\text{out}} \times r}$. The cluster stores its pooled basis $\tilde{U}_c$ and singular values Σ_c once. We approximate the decomposition using the Gaussian-sketch power iteration with rank-one deflation adopted by TileQ (Gu et al., 2026a), avoiding an exact SVD of the full $d_{\text{in}} \times G_c d_{\text{out}}$ matrix. Expert k retains its activation scale s_k and the corresponding row block $V_k \in \mathbb{R}^{d_{\text{out}} \times r}$ of $\tilde{V}_c$. Its correction is

$$\Delta_k \;=\; \text{diag}(s_k)^{-1}\, \tilde{U}_c\, \Sigma_c\, V_k^{\top}. \tag{6}$$

We then quantize the residual:

$$Q_k \;=\; \mathcal{Q}\big(\hat{W}_k - \Delta_k\big), \qquad \hat{W}_k \;\approx\; Q_k + \Delta_k. \tag{7}$$

The storage tradeoff can be stated explicitly. Let $M = LN$ be the number of experts of one weight type, H the number of pooled bases, b_U and b_V the stored bits per factor element, and B_{scale} the storage for activation and factor scales. The correction cost and the largest rank supported by a correction budget B are approximately

$$B_{\text{corr}}(r, H) = r\big(b_U H d_{\text{in}} + b_V M d_{\text{out}}\big) + B_{\text{scale}}, \qquad r_{\max} = \left\lfloor \frac{B - B_{\text{scale}}}{b_U H d_{\text{in}} + b_V M d_{\text{out}}} \right\rfloor. \tag{8}$$

Private corrections correspond to $H = M$, whereas pooling G experts per group uses about $H = \lceil M/G \rceil$ bases. Each expert still stores V_k and s_k, so pooling reduces the shared-basis term rather than the entire correction cost. The released basis budget can be assigned to a larger r.

Cross-layer grouping also enlarges the set of admissible partitions at the same storage budget. Let $\mathcal{P}_{\text{layer}}$ contain size-bounded partitions whose groups stay within one layer, and let $\mathcal{P}_{\text{cross}}$ allow the same number and sizes of groups across all layers. For a partition $\mathcal{P}$, define the activation-weighted rank-r fitting loss

$$\mathcal{J}(\mathcal{P}) = \sum_{c \in \mathcal{P}} \|A_c - [A_c]_r\|_F^2, \qquad A_c = [\,\text{diag}(s_k)\hat{W}_k\,]_{k \in c}, \tag{9}$$

where $[A_c]_r$ is the best rank-r approximation. Since $\mathcal{P}_{\text{layer}} \subseteq \mathcal{P}_{\text{cross}}$,

$$\min_{\mathcal{P} \in \mathcal{P}_{\text{cross}}} \mathcal{J}(\mathcal{P}) \;\le\; \min_{\mathcal{P} \in \mathcal{P}_{\text{layer}}} \mathcal{J}(\mathcal{P}). \tag{10}$$

Thus cross-layer pooling can select a lower-error shared fit without adding groups or changing the correction rank. The Grassmannian clustering is a practical way to search this larger partition set.

Appendix B derives the activation-weighted objective and its cluster-shared solution. Appendix C expands Eq. (8) and separates the reported nominal bit rate from the auxiliary tensors in the exported representation.

Pooling works best when the grouped experts have similar active subspaces. If their dominant directions differ, the joint rank-r approximation must divide its capacity among them. The distance in Eq. (2) is used to identify compatible subspaces before pooling. Section 5.4 compares this grouping with random assignment and varies the group size.

Algorithm 1 in Appendix A summarizes the procedure. A calibration pass collects activation statistics; clustering, factor fitting, and residual quantization follow without finetuning. The pooled and private factors use the same low-bit representation. Attention weights are quantized separately with GPTQ (Frantar et al., 2022) and do not receive a CLASP correction.

4.3 Inference-Side Co-Design

On the decode traces in Figure 3, Qwen1.5-MoE-A2.7B's four routed experts span 1.73 clusters on average, while Qwen3-30B-A3B's eight span 3.9 of 48 clusters. This concentration appears in the gate and up projections, but not in the down projection. We use a device-resident factor pool to avoid storing a private copy of every basis and batch correction operations to reduce launch overhead. When beneficial, the correction path runs concurrently with the backbone.

Active-subspace pool. Each cluster's factors are stored once in a pool shared across layers. An active expert accesses $(\tilde{U}_c, \Sigma_c)$ together with its private V_k and scale s_k. Concentration within a few clusters reduces the number of distinct pooled bases needed for a token (Figure 3(a)). These factors reside in device memory; their reuse may also improve cache locality.

Batched correction. For the gate or up projection, let $x \in \mathbb{R}^{1 \times d_{\text{in}}}$ be the input shared by the k routed experts $e_1, \ldots, e_k$. For expert e, let $c(e)$ denote its cluster. The correction output can be written as

$$x\,\Delta_e \;=\; z_e\,V_e^\top, \tag{11}$$

$$z_e \;=\; x\,\text{diag}(s_e)^{-1}\,\tilde{U}_{c(e)}\,\Sigma_{c(e)} \;\in\; \mathbb{R}^{1 \times r}, \tag{12}$$

where z_e is a rank-r row vector. The expert-specific scale s_e means that experts in the same cluster do not generally share z_e, even though they share the stored basis. Small per-expert matrix-vector

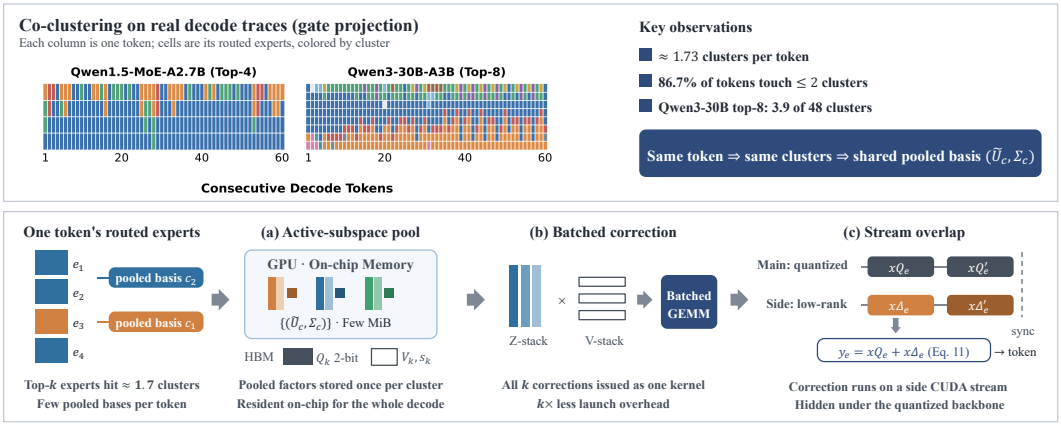

Figure 3: Factor sharing and correction execution. **Top:** decode traces for the gate projection. Each column is a token, with routed experts colored by cluster. Experts with the same color use the same pooled basis $\tilde{U}_c$. **Bottom:** the factor pool, batched correction operations, and stream overlap. The pool is stored in device memory and may benefit from cache reuse; persistent on-chip residency is not assumed.

products can incur substantial launch overhead during decoding. We batch the second-stage products across experts,

$$\left[\, x\Delta_{e_1};\ \cdots\ ;\ x\Delta_{e_k} \,\right] \;=\; \left[\, z_{e_1} V_{e_1}^{\top};\ \cdots\ ;\ z_{e_k} V_{e_k}^{\top} \,\right] \;\in\; \mathbb{R}^{k \times d_{\text{out}}}, \tag{13}$$

reducing separate launches for these products (Figure 3(b)). Forming the z_e vectors is also batched where possible. The down projection uses the same formulation with an expert-specific input x_e, since each expert produces a different intermediate activation.

Stream overlap. For each linear projection, the backbone and correction products depend on the same input and can be evaluated independently:

$$y_e \;=\; \underbrace{x\,Q_e}_{\text{main stream}} \;+\; \underbrace{x\,\Delta_e}_{\text{side stream}}, \tag{14}$$

Here y_e is the output of one linear projection, rather than the complete nonlinear expert. The backbone product can run on the main CUDA stream while the correction runs on a side stream (Figure 3(c)). The streams synchronize before the two results are combined for the next dependent operation. After the down projection, the expert outputs are combined using the router weights. CUDA-graph capture reduces the scheduling overhead of this execution pattern; Section 5.3 measures its contribution.

5 Experiments

5.1 Setup

Models and configuration. We evaluate Qwen1.5-MoE-A2.7B, Mixtral-8x7B, Qwen3-30B-A3B, and DeepSeek-MoE-16B. Quantization runs on a single GPU without finetuning, using 128 WikiText-2 calibration sequences (Merity et al., 2016). The routed-expert backbone $\mathcal{Q}(\cdot)$ uses two-bit vector quantization (Gu et al., 2026a). We use $G = 128$ and activation-scale exponent $\alpha = 2.4$; Mixtral uses $G = 8$. The correction rank is $r = 32$ for Qwen1.5 and Mixtral, and $r = 16$ for Qwen3 and DeepSeek. The randomized SVD used to extract clustering subspaces runs four power iterations; spectral clustering uses five k-means initializations with seed 42. The pooled decomposition uses up to eight rank-one sketch iterations. All four models use four-bit GPTQ for attention, without a low-rank correction, and the same INT8 representation for the pooled and private correction factors. DeepSeek's shared experts and dense first layer are also quantized to four bits. The complete representations use 2.15–2.21 bits per quantized parameter.

Model	Method	Bits	PPL↓	ARC-C	ARC-E	PIQA	WinoG	HellaS	Avg.
Qwen1.5-MoE-A2.7B	FP16	16	6.79	41.6	72.9	79.6	69.14	59.3	64.50
	GPTQ	3.13	7.98	33.4	64.3	77.1	64.5	52.2	58.30
	MoQE	3.00	8.21	32.6	63.6	76.2	63.8	50.4	57.32
	LoPRo	2.43	7.52	**39.9**	72.7	77.6	68.2	53.4	62.36
	TileQ	2.16	7.56	39.6	72.5	77.8	68.9	**55.5**	**62.86**
	GPTQ	2.13	12.5	29.4	42.7	50.2	53.2	30.1	41.12
	CLASP (ours)	2.15	**7.14**	39.68	**72.73**	**78.02**	**69.10**	54.24	62.76
Mixtral-8x7B	FP16	16	3.87	61.9	87.3	83.7	77.1	67.3	75.46
	GPTQ	3.13	4.72	52.4	69.3	79.1	74.4	44.8	64.00
	MoQE	3.00	5.45	**57.4**	80.2	78.9	71.9	60.1	69.70
	LoPRo	2.21	5.01	55.3	82.5	80.6	74.9	**60.3**	70.72
	TileQ	2.16	4.98	55.5	**82.8**	**80.9**	**75.1**	**60.3**	**70.92**
	GPTQ	2.13	15.3	26.5	35.6	53.0	49.3	28.2	38.52
	CLASP (ours)	2.16	**4.50**	48.12	78.45	78.18	75.06	55.62	67.09
Qwen3-30B-A3B	FP16	16	8.07	52.6	79.2	79.7	70.3	58.8	68.12
	GPTQ	3.13	9.42	44.0	70.3	75.1	63.4	50.1	60.58
	MoQE	3.00	28.1	37.8	30.6	59.4	55.3	43.2	45.26
	LoPRo	2.58	11.1	34.4	58.3	71.5	62.9	48.0	55.02
	TileQ	2.16	11.3	34.6	58.4	71.8	63.1	48.3	55.24
	GPTQ	2.13	14.6	31.1	54.4	68.9	57.2	43.2	50.96
	CLASP (ours)	2.16	**8.97**	**45.65**	**74.41**	**76.33**	**67.56**	**53.36**	**63.46**
DeepSeek-MoE-16B	FP16	16	6.11	45.1	75.9	78.6	70.5	59.7	65.96
	GPTQ	3.13	7.17	32.3	66.7	**77.1**	67.9	**56.9**	60.18
	MoQE	3.00	7.78	37.6	69.2	74.3	62.1	45.8	57.80
	LoPRo	2.58	7.03	38.3	72.5	75.8	68.4	52.5	61.50
	TileQ	2.16	7.06	38.1	**72.7**	76.1	68.5	52.2	61.52
	GPTQ	2.13	24.1	24.4	50.2	66.5	53.3	45.3	47.94
	CLASP (ours)	2.21	**6.60**	**38.57**	72.43	76.77	**68.98**	52.91	**61.93**

Table 1: Perplexity and five-task zero-shot accuracy (%) on four MoE models. Bits reports the average nominal bits per quantized parameter; PPL is WikiText-2 perplexity. ARC-C/E, WinoG, and HellaS denote ARC-Challenge/Easy, WinoGrande, and HellaSwag. Avg. is the five-task mean. Bold marks the best quantized result, including ties.

Bit accounting. We report nominal bits divided by the number of quantized parameters, excluding embeddings and the LM head. This accounting includes the vector-quantization code payload, four-bit attention weights, and INT8 correction-factor values. Each cluster contributes one pooled basis, while each expert contributes a private factor. Appendix C gives the formula used for the reported rate and separately itemizes the FP16 scales, activation scales, codebooks, and transform metadata in the real-quant export.

Baselines. We compare with GPTQ (Frantar et al., 2022), MoQE (Kim et al., 2023), LoPRo (Gu et al., 2026b), and TileQ (Gu et al., 2026a). All are evaluated as weight-quantization baselines without finetuning. Their reported bit budgets are listed in Table 1.

Comparison protocol. Table 1 compares accuracy and storage at the reported bit budgets. For TileQ, no public end-to-end implementation was available at evaluation time, so we use its reported accuracy and bit rate. A cross-paper latency ratio would also mix kernels, hardware, and timing protocols. The systems study compares the optimized CLASP implementation with FP16 under the same workload, measurement protocol, and GPU; Table 2 separates batching, graph capture, and stream overlap.

5.2 Accuracy

Table 1 reports WikiText-2 perplexity and zero-shot accuracy on ARC-Challenge and ARC-Easy (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), and HellaSwag (Zellers et al., 2019), evaluated with lm-eval (Gao et al., 2023). CLASP has the lowest perplexity among the compared quantizers on all four models. On Qwen3-30B-A3B, it improves the

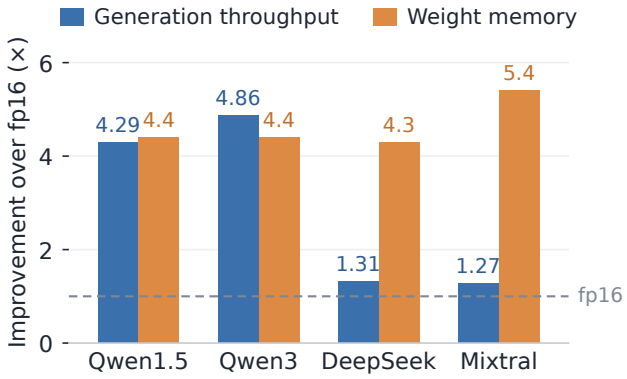

Figure 4: End-to-end generation throughput at batch size 1 and resident weight memory reduction relative to the FP16 baseline under matched workload, timing protocol, and hardware. Throughput ratios are CLASP/FP16; memory reduction factors are FP16/CLASP.

Decode configuration	**tokens/s**	**Δ**
per-expert corrections, serial	10.8	–
+ batched formulation	20.4	+89%
+ CUDA-graph capture	21.4	+5%
+ side-stream overlap (full)	21.8	+2%
full pipeline − batched corrections	12.7	−41%

Table 2: Decode-mechanism ablation on Qwen1.5-MoE-A2.7B (2.15 bits; batch size 1; A100). Δ is relative to the preceding row for rows 2–4 and to the full pipeline for the last row. Rows 1–4 are cumulative; the last row removes only the batched corrections from the full pipeline. All configurations emit identical greedy token IDs.

five-task average over TileQ by 8.22 percentage points and reaches a perplexity of 8.97, compared with 8.07 for FP16. On DeepSeek-MoE-16B, it improves both perplexity and average accuracy over TileQ, with a slightly larger bit budget. On Qwen1.5-MoE-A2.7B, it lowers perplexity from 7.56 to 7.14, while the average accuracy is 0.10 points lower (62.76 vs. 62.86).

The largest improvement over TileQ occurs on Qwen3-30B-A3B, the model with the most routed experts in this comparison. The outcome is less favorable on Mixtral: CLASP lowers perplexity from 4.98 to 4.50, but its zero-shot average falls from 70.92 to 67.09. Pooling therefore does not improve every metric or architecture. The controlled clustering ablation in Section 5.4 provides more direct evidence for the value of subspace-based grouping.

5.3 Inference Efficiency

Figure 4 compares the optimized CLASP implementation with FP16 under the same workload, measurement protocol, and GPU. End-to-end timing covers prefill and decode at batch size 1 with 128 prompt and 128 generated tokens. Throughput increases by 4.3×–4.9× on the Qwen models and 1.27×–1.31× on Mixtral and DeepSeek; resident weight memory falls by 4.3×–5.4×. Mixtral uses 512 generated tokens in the figure and reaches 1.24× at 128 tokens.

Speedup also reflects dispatch structure. The Qwen3 baseline visits 6,144 expert entries per token, while DeepSeek skips unselected experts and makes 162 active-expert visits, leaving more per-expert overhead to reduce on Qwen.

Execution ablation. Table 2 isolates the execution changes on Qwen1.5-MoE-A2.7B. Batching raises throughput from 10.8 to 20.4 tokens/s, the largest gain. Without batching, the full pipeline falls from 21.8 to 12.7 tokens/s. Graph capture and stream overlap raise throughput to 21.4 and 21.8 tokens/s. Batching accounts for most of the correction-path speedup.

The execution and representation results answer different parts of the deployment question. Cross-layer grouping determines which bases can be shared and reduces their storage. Batching supplies most of the measured execution gain once those corrections are present. The end-to-end speedup combines the low-bit backbone with this optimized correction path.

5.4 Ablations

r	G	**Bits**	**PPL↓**
Rank sweep ($G = 128$)			
16	128	2.08	7.845
32	128	2.15	7.138†
64	128	2.29	7.087
Cluster-size sweep ($r = 32$)			
32	32	2.16	7.220
32	64	2.15	7.103
No correction			
0	–	2.01	7.920

Table 3: Ablations on Qwen1.5-MoE-A2.7B. † marks the default, which also supplies the $G = 128$ point in the cluster-size sweep.

Model	**Grassmannian**	**Random**	**ΔPPL**
Qwen1.5-MoE-A2.7B	**7.14**	7.44	+0.30
Qwen3-30B-A3B	**8.97**	9.83	+0.86

Table 4: Grassmannian versus random expert-to-cluster assignment at matched budget and cluster size.

Rank and cluster size. Table 3 varies rank and cluster size on Qwen1.5-MoE-A2.7B. Removing the correction saves 0.14 bits but raises perplexity from 7.14 to 7.92. Increasing r from 16 to 32 reduces perplexity from 7.85 to 7.14; increasing it to 64 gives a smaller reduction, to 7.09, while adding 0.14 bits. We use $r = 32$ for this model. At that rank, cluster sizes of 32, 64, and 128 give perplexities within 0.12 at nearly equal bit budgets. The default $G = 128$ groups the $1{,}440$ expert matrices of each weight type into 12 shared bases, including one smaller final group.

Subspace-based versus random grouping. Table 4 compares Grassmannian clustering with random assignment at the same budget and group size. Random assignment increases perplexity by 0.30 on Qwen1.5-MoE-A2.7B and 0.86 on Qwen3-30B-A3B. On both evaluated Qwen models, grouping experts by active-subspace similarity improves the shared-basis fit over random grouping. Because rank, group size, and storage are fixed, this comparison isolates the assignment rule rather than additional correction capacity.

6 Conclusion

CLASP is motivated by two empirical observations. First, activation-weighted expert subspaces overlap across layers: on Qwen3-30B-A3B, the nearest subspace neighbor of 71% of experts lies in another layer. Second, experts activated by the same token tend to concentrate in a small number of the resulting clusters on the evaluated Qwen models. CLASP uses this structure to share correction bases across layers, enlarging the partition space at a fixed basis budget. At 2.15–2.21 bits per quantized parameter, it achieves the lowest perplexity among the compared quantizers on all four models; the TileQ entries use published accuracy and storage results. In the matched-protocol systems study, the optimized correction path reduces resident weight memory by $4.3\times$–$5.4\times$ and improves end-to-end throughput by up to $4.9\times$ over FP16.

AI USE STATEMENT

We used generative AI tools to assist with drafting and revising the manuscript and with preparing figures based on author-provided data, results, and design specifications. The authors determined the research questions, methodology, experimental setup, and interpretation of the results. All AI-generated text and visual content was reviewed, edited, and verified by the authors. The authors take full responsibility for the accuracy, citations, figures, and conclusions of the final manuscript.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328, 2021.

Selim An, Il hong Suh, and Yeseong Kim. GlowQ: Group-Shared LOw-Rank Approximation for Quantized LLMs. *arXiv preprint arXiv:2603.25385*, 2026.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: outlier-free 4-bit inference in rotated LLMs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc.

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

Åke Björck and Gene H. Golub. Numerical Methods for Computing Angles Between Linear Subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.

Shiyi Cao, Shu Liu, Tyler Griggs, Peter Schafhalter, Xiaoxuan Liu, Ying Sheng, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. MoE-Lightning: High-Throughput MoE Inference on Memory-constrained GPUs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 715–730, 2025.

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

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)*, pp. 1280–1297, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: efficient finetuning of quantized LLMs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Tim Dettmers, Ruslan A Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *The twelfth international conference on learning representations*, 2024.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. GLaM: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569, 2022.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Artyom Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading. *arXiv preprint arXiv:2312.17238*, 2023.

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.

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.

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.

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. URL `https://zenodo.org/records/10256836`.

Hongyaoxing Gu, Xinzhe Chen, Lijuan Hu, and Fangfang Liu. TileQ: Efficient Low-Rank Quantization of Mixture-of-Experts with 2D Tiling. *arXiv preprint arXiv:2605.09281*, 2026a.

Hongyaoxing Gu, Lijuan Hu, Liye Yu, Haowei Li, and Fangfang Liu. LoPRo: Enhancing Low-Rank Quantization via Permuted Block-Wise Rotation. *arXiv preprint arXiv:2601.19675*, 2026b.

Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. LQ-LoRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024.

Jihun Hamm and Daniel D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 376–383, 2008.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

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

Beichen Huang, Yueming Yuan, Zelei Shao, and Minjia Zhang. MiLo: Efficient quantized MoE inference with mixture of low-rank compensators. In *Proceedings of Machine Learning and Systems*, volume 7, 2025.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts LLMs gains more. *arXiv preprint arXiv:2410.06270*, 2024.

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.

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.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024.

Geonho Lee, Janghwan Lee, Sukjin Hong, Minsoo Kim, Euijai Ahn, Du-Seong Chang, and Jungwook Choi. RILQ: Rank-insensitive LoRA-based quantization error compensation for boosting 2-bit large language model accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18091–18100, 2025.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024a.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient SMoE with hints from its routing policy. *arXiv preprint arXiv:2310.01334*, 2023.

Pingzhi Li, Xiaolong Jin, Zhen Tan, Yu Cheng, and Tianlong Chen. QuantMoE-Bench: Examining Post-Training Quantization for Mixture-of-Experts. *arXiv preprint arXiv:2406.08155*, 2024b.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. MoE-SVD: Structured mixture-of-experts LLMs compression via singular value decomposition. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35209–35230. PMLR, 2025.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, 2024c.

Baohao Liao, Christian Herold, Shahram Khadivi, and Christof Monz. ApiQ: Finetuning of 2-bit quantized large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20996–21020, 2024.

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 LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. VPTQ: Extreme low-bit vector post-training quantization for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8181–8196, 2024.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172, 2024.

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

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, pp. 849–856, 2001.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International conference on machine learning*, pp. 18332–18346, 2022.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8694–8705, 2024.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J. Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8732–8740, 2020.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Peng Tang, Jiacheng Liu, Xiaofeng Hou, Yifei Pu, Jing Wang, Pheng-Ann Heng, Chao Li, and Minyi Guo. Hobbit: A mixed precision expert offloading system for fast MoE inference. *arXiv preprint arXiv:2411.01433*, 2024.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: even better LLM quantization with Hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. *arXiv preprint arXiv:2410.03765*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

He Xiao, Runming Yang, Qingyao Yang, Wendong Xu, Zhen Li, Yupeng Su, Zhengwu Liu, Hongxia Yang, and Ngai Wong. PTQTP: Post-training quantization to trit-planes for large language models. *arXiv preprint arXiv:2509.16989*, 2025.

He Xiao, Qingyao Yang, Dirui Xie, Wendong XU, Zunhai Su, Runming Yang, Haobo Liu, Wenyong Zhou, Zhengwu Liu, and Ngai Wong. Exploring layer-wise information effectiveness for post-training quantization in small language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15753–15764, 2026. doi: 10.18653/v1/2026.findings-acl.771.

Yuebin Xu, Yanhong Wang, Xuemei Peng, Hui Zang, Minghao Chen, Pengfei Xia, and Zeyi Wen. TD-MoE: Tensor decomposition for MoE models. In *International Conference on Learning Representations*, 2026a.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. KBVQ-MoE: Klt-guided SVD with bias-corrected vector quantization for MoE large language models. *arXiv preprint arXiv:2602.11184*, 2026b.

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. ZeroQuant-V2: Exploring post-training quantization in LLMs from comprehensive study to low rank compensation. *arXiv preprint arXiv:2303.08302*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. LQER: low-rank quantization error reconstruction for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Cheng Zhang, Jeffrey TH Wong, Can Xiao, George Constantinides, and Yiren Zhao. QERA: an analytical framework for quantization error reconstruction. In *International Conference on Learning Representations*, 2025.

A THE ONE-SHOT QUANTIZATION PROCEDURE

Algorithm 1 CLASP one-shot quantization

Input: expert weights $\{\hat{W}_i\}_{i=1}^{LN}$ per weight type; calibration set $\mathcal{D}$; rank r; cluster size G (giving $C = \lceil LN/G \rceil$)
Output: backbone codes $\{Q_i\}$; pooled bases $\{\tilde{U}_c\}_{c=1}^{C}$; private folded factors $\{T_i\}$; scales $\{s_i\}$

$s_i \leftarrow$ per-channel activation scales on $\mathcal{D}$, for every expert i
for each weight type **do**
  for $i = 1$ **to** LN **do**
    $U_i \leftarrow$ active subspace: top-r left singular basis of $\mathrm{diag}(s_i)\,\hat{W}_i$
  end for
  $d(i,j) \leftarrow$ Eq. (2) for all pairs (i,j); $A_{ij} \leftarrow$ Eq. (3)
  $\{\mathcal{C}_c\}_{c=1}^{C} \leftarrow$ spectral clustering of A via Eq. (4), ordered by cluster label and split into groups of at most G experts
  for $c = 1$ **to** C **do**
    $\tilde{U}_c \Sigma_c \tilde{V}_c^\top \leftarrow$ sketch-based SVD_r of the stacked matrix in Eq. (5) {pool the cluster's active subspaces}
    $V_k \leftarrow k$-th row block of $\tilde{V}_c$, $\forall k \in \mathcal{C}_c$
    $T_k \leftarrow V_k \Sigma_c$, $\forall k \in \mathcal{C}_c$; store $\tilde{U}_c$ and $\{T_k\}_{k \in \mathcal{C}_c}$ in INT8 with FP16 scale metadata
  end for
  for each expert k **do**
    $\Delta_k \leftarrow \mathrm{diag}(s_k)^{-1} \tilde{U}_{c(k)} T_k^\top$
    $Q_k \leftarrow \mathcal{Q}\big(\hat{W}_k - \Delta_k\big)$ {backbone quantizer}
  end for
end for

B ACTIVATION-WEIGHTED LOW-RANK RECONSTRUCTION

This section makes explicit the surrogate objective behind the active-subspace construction. For one expert, let $x \in \mathbb{R}^{1 \times d_{\text{in}}}$ be a calibration input and let $E = \hat{W} - \Delta$ denote the weight error after applying a rank-r correction. The expected squared output error is

$$\mathbb{E}_x\big[\|xE\|_2^2\big] = \mathrm{tr}\big(E^\top C_x E\big), \qquad C_x = \mathbb{E}_x[x^\top x]. \tag{B.1}$$

Using the full covariance C_x would couple all input channels and make calibration and factor fitting more expensive. CLASP instead uses the diagonal activation statistic described in Section 4.2. Writing $S = \mathrm{diag}(s)$, the resulting surrogate is

$$\mathrm{tr}\big(E^\top S^2 E\big) = \|SE\|_F^2. \tag{B.2}$$

The statistic $s_q \propto \mathbb{E}[|x_q|]^\alpha$ is a robust channel-importance heuristic rather than an exact covariance estimate. The geometric-range normalization controls its scale, while clamping prevents nearly inactive channels from producing unstable inverse scales.

For a single expert, the rank-constrained problem induced by Eq. (B.2) is

$$\min_{\mathrm{rank}(\Delta) \le r} \Big\| S\hat{W} - S\Delta \Big\|_F^2. \tag{B.3}$$

Let $A = S\hat{W}$. Since S is invertible after clamping, $B = S\Delta$ has the same rank as Δ. If $A = U\Sigma V^\top$, the Eckart–Young theorem gives the minimizer $B^\star = U_r \Sigma_r V_r^\top$ and hence

$$\Delta^\star = S^{-1} U_r \Sigma_r V_r^\top. \tag{B.4}$$

This is the single-expert form of Eq. (6).

For a cluster $\mathcal{C}_c$, define $A_k = S_k \hat{W}_k$ and concatenate the matrices as $A_c = [A_k]_{k \in \mathcal{C}_c}$. A rank-$r$ approximation of A_c solves

$$\min_{\tilde{U}_c, \Sigma_c, \{V_k\}} \sum_{k \in \mathcal{C}_c} \Big\| A_k - \tilde{U}_c \Sigma_c V_k^\top \Big\|_F^2 \quad \text{s.t.} \quad \tilde{U}_c^\top \tilde{U}_c = I_r. \tag{B.5}$$

Indeed, partitioning the right singular vectors of A_c into row blocks recovers the V_k in Eq. (B.5). The truncated SVD of A_c is therefore the optimal shared rank-r approximation under this surrogate. CLASP uses a sketch-based approximation to avoid materializing an exact decomposition of the full concatenated matrix.

After fitting the correction, the backbone quantizer acts on the residual $R_k = \hat{W}_k - \Delta_k$. The reconstructed weight and its remaining error can be written as

$$\hat{W}_k^{\mathrm{rec}} = \mathcal{Q}(R_k) + \Delta_k, \qquad \hat{W}_k - \hat{W}_k^{\mathrm{rec}} = R_k - \mathcal{Q}(R_k). \tag{B.6}$$

Thus the low-rank component removes selected directions before the low-bit backbone is formed; quantization only needs to represent the residual that remains.

B.1 Why the Grassmannian distance is basis independent

Each active subspace admits many orthonormal bases. If $R_i, R_j \in \mathbb{R}^{r\times r}$ are orthogonal, replacing U_i and U_j by U_iR_i and U_jR_j gives

$$(U_iR_i)^\top(U_jR_j) = R_i^\top(U_i^\top U_j)R_j. \tag{B.7}$$

Left and right multiplication by orthogonal matrices leaves singular values unchanged. The principal angles and the distance in Eq. (2) therefore depend only on the two subspaces, not on the particular singular-vector bases returned by the numerical routine. Since every principal angle lies in $[0, \pi/2]$, the normalization by $\sqrt{r}\,\pi/2$ also places $d(i, j)$ in $[0, 1]$.

C Detailed Bit and Storage Accounting

We use two related quantities. The nominal bit rate reported in the main tables counts vector codes and quantized factor values. The real-quant export additionally contains the scales, codebooks, transform metadata, and cluster assignments needed for decoding. Both are independent of container headers, alignment padding, tokenizer files, and other non-weight artifacts. Let one routed weight type contain M expert matrices in H clusters. Each matrix has shape $d_{\mathrm{in}} \times d_{\mathrm{out}}$. In the exported representation, Σ_c is folded into the private factor

$$T_k = V_k\Sigma_c, \qquad \Delta_k = S_k^{-1}\tilde{U}_cT_k^\top, \tag{C.1}$$

so no separate singular-value tensor is stored.

Correction factors. For the `int8` representation, the factor-value term counted in the reported rate is

$$B^{(8)}_{\mathrm{corr,val}} = 8r(Hd_{\mathrm{in}} + Md_{\mathrm{out}}). \tag{C.2}$$

Each cluster also stores r FP16 basis scales. Each expert stores r FP16 private-factor scales and an FP16 activation-scale vector of length d_{in}. Including the logical cluster assignment, these auxiliary terms are

$$B^{(8)}_{\mathrm{corr,aux}} = 16r(H + M) + 16Md_{\mathrm{in}} + M\left\lceil \log_2 H \right\rceil. \tag{C.3}$$

Equation (8) uses the value term and can include Eq. (C.3) through B_{scale} when deployment payload rather than nominal rate is required. For an average group size $\bar{G} = M/H$, the exported INT8 correction payload per expert matrix parameter is

$$\beta^{(8)}_{\mathrm{corr,export}} = \frac{8r(d_{\mathrm{in}}/\bar{G} + d_{\mathrm{out}}) + 16r(1/\bar{G} + 1) + 16d_{\mathrm{in}} + \lceil \log_2 H \rceil}{d_{\mathrm{in}}d_{\mathrm{out}}}. \tag{C.4}$$

This expression shows directly which term pooling reduces: $d_{\mathrm{in}}r$ basis values and their scales are amortized over $\bar{G}$ experts, whereas the private factor and activation scales remain per expert.

Vector-quantized backbone. Suppose each code selects one of K vectors of dimension v. For expert k, the code payload is

$$B_{\mathrm{code},k} = \frac{d_{\mathrm{in}}d_{\mathrm{out}}}{v}\log_2 K. \tag{C.5}$$

Our two-bit backbone uses $K = 256$ and $v = 4$, so each four-weight vector has one 8-bit index and Eq. (C.5) equals $2d_{\mathrm{in}}d_{\mathrm{out}}$ bits. The real-quant export also stores FP16 codebooks, an INT32 input-channel permutation, and FP16 diagonal signs. If expert k has $n_{\mathrm{cb},k}$ codebook blocks, the auxiliary backbone payload is

$$B_{\mathrm{VQ,aux},k} = 16n_{\mathrm{cb},k}Kv + 32d_{\mathrm{in}} + 16d_{\mathrm{in}}. \tag{C.6}$$

Model-level average. Let $\mathcal{T}$ index the routed projection types. The nominal routed-expert term is

$$B_{\text{routed,nom}} = \sum_{t \in \mathcal{T}} \left(\sum_{k=1}^{M_t} B_{\text{code},t,k} + B_{\text{corr,val},t} \right). \tag{C.7}$$

The reported model-level value is therefore

$$\text{Bits} = \frac{B_{\text{routed,nom}} + B_{\text{attn}} + B_{\text{shared}} + B_{\text{dense}}}{P_{\text{quant}}}, \tag{C.8}$$

where P_{quant} is the number of parameters in the declared quantized scope. B_{attn} counts four-bit GPTQ attention for every model and contains no correction term. B_{shared} and B_{dense} are nonzero for the four-bit shared experts and dense first layer in DeepSeek. Embeddings and the language-model head are outside both the numerator and denominator. The tensor payload of the exported routed weights is obtained by adding $\sum_t B_{\text{corr,aux},t} + \sum_{t,k} B_{\text{VQ,aux},t,k}$ to the numerator. This second quantity is useful when provisioning device memory for a concrete checkpoint.

D Batched Correction at Decode Time

Algorithm 2 describes one routed linear projection during decoding. The projection may be the gate, up, or down matrix; for the down projection, x_e denotes the expert-specific intermediate activation. Experts can belong to different clusters, so the implementation gathers the required pooled bases before issuing the batched matrix multiplications.

Algorithm 2 Batched low-rank correction for one routed projection

Input: active experts $\mathcal{E}$; inputs $\{x_e\}$; router weights $\{\rho_e\}$; backbones $\{Q_e\}$; assignments $c(e)$; factors $\{\tilde{U}_c, T_e, S_e^{-1}\}$
Output: routed projection output y

1: launch $b_e \leftarrow x_e Q_e$ for all $e \in \mathcal{E}$ on the main stream
2: gather $\tilde{U}_{c(e)}$, T_e, and S_e^{-1} for the active experts
3: $z_e \leftarrow (x_e S_e^{-1}) \tilde{U}_{c(e)}$ for all $e \in \mathcal{E}$ using one batched matrix multiplication
4: $\delta_e \leftarrow z_e T_e^\top$ for all $e \in \mathcal{E}$ using one batched matrix multiplication
5: synchronize the backbone and correction streams
6: $y_e \leftarrow b_e + \delta_e$ for all $e \in \mathcal{E}$
7: $y \leftarrow \sum_{e \in \mathcal{E}} \rho_e y_e$ {applied after the down projection}