SAGE-VQ: STRUCTURE-ADAPTIVE GATING-AWARE EXPERT VECTOR QUANTIZATION WITH OUTPUT CORRECTION FOR MOE LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) large language models (LLMs) expand capacity through sparse activation, yet expert storage remains a bottleneck for single-GPU and on-device deployment. Post-training vector quantization (VQ) enables ultra-low-bit compression, but dense-LLM methods overlook heterogeneous expert sensitivity, architecture-dependent output composition, and routing-weighted errors. We introduce *SAGE-VQ*, a structure-adaptive, gating-aware expert VQ framework that aligns precision allocation and output correction with MoE structure. *SAGE-VQ* selects architecture-appropriate sensitivity signals to assign expert bit-widths under a fixed average bit budget. It incorporates routing weights into the correction objective to mitigate norm and scale mismatches in quantized expert outputs. Correction factors are fused before inference without additional runtime latency. On Qwen2-MoE, *SAGE-VQ* achieves 2.25-bit expert compression, reducing WikiText-2 perplexity by 1.21 relative to the MoE-aware mixed-precision baseline MxMoE while limiting average accuracy degradation across seven zero-shot benchmarks to 1.42 percentage points relative to FP16.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance across a wide range of tasks (Brown et al., 2020; Touvron et al., 2023). However, scaling dense LLMs increases both computation and memory footprint with the number of parameters, resulting in higher deployment costs. Mixture-of-Experts (MoE) models mitigate this growth in computational cost by routing each token to only a subset of experts, thereby limiting the active computation per token while expanding model capacity (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Owing to this capacity–computation trade-off, MoE architectures have been adopted in recent LLMs such as Mixtral (Jiang et al., 2024), Qwen-MoE (Yang et al., 2024), and DeepSeek-MoE (Dai et al., 2024; Liu et al., 2024a).

Despite their sparse computation, MoE models still require substantial memory. Because the router can select different experts for each token, the parameters of all experts must either remain in accelerator memory or be repeatedly fetched during inference. For example, Qwen2-MoE activates only about 14 B parameters for each token, but contains 57 B parameters in total, requiring approximately 114 GB of memory to store its weights in FP16. This large memory footprint makes it difficult to serve MoE LLMs on a single GPU or edge device, motivating the reduction of expert parameter storage for deployment in memory-constrained environments (Yi et al., 2025; Yang et al., 2026b; Xue et al., 2024).

Post-training quantization (PTQ) is widely used to compress LLMs because it reduces the memory footprint of pretrained models without costly retraining (Frantar et al., 2023; Lin et al., 2024; Xiao et al., 2023). Most PTQ methods rely on scalar quantization (SQ), which maps individual weight values to a low-precision grid using scales defined at the tensor, channel, or group level. However, in the ultra-low-bit (ULB) regime, particularly below 3 bits, the number of discrete scalar values available to represent each weight decreases sharply, which can substantially increase quantization error. In contrast, vector quantization (VQ) represents the joint pattern of multiple weights with a single codeword rather than representing each weight independently with one of a limited number

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.

of scalar values. This allows weight combinations to be represented more flexibly under the same bit budget and to be better matched to the data distribution, alleviating the limited representational capacity of SQ in the ULB regime. Recent VQ-based methods have shown promising accuracy–compression trade-offs for ULB compression of dense LLMs (Egiazarian et al., 2024; Tseng et al., 2024a;b).

Directly applying VQ methods developed for dense LLMs to MoE models overlooks key structural characteristics of MoE architectures. In an MoE model, the router assigns different tokens to different experts, resulting in distinct input distributions and usage frequencies across experts. Consequently, even under the same quantization scheme, the quantization error of each expert can affect the MoE block output differently. This motivates expert-wise mixed-precision quantization that allocates higher precision to more important experts. Moreover, MoE architectures differ in their gating mechanisms and expert-output aggregation, so the allocation signal that best reflects expert importance may also vary across architectures. Applying a single fixed allocation signal across different MoE structures can therefore lead to sub-optimal expert-wise bit allocation.

ULB VQ can also introduce distortion and scale mismatch in expert outputs. Because routed expert outputs are weighted by token-specific routing weights before aggregation, the same expert-local error can have different effects on the MoE block output. Output correction should therefore account for routing weights in addition to expert-local errors.

To address these challenges, we propose *SAGE-VQ*, a Structure-Adaptive Gating-Aware Expert Vector Quantization framework for ULB MoE compression. First, the structure-adaptive allocation policy (SAP) selects an allocation signal that reflects expert importance according to the MoE output topology, and mixed vector quantization (MVQ) uses the selected signal to assign different VQ bit-widths to experts under the same average bit budget. Second, routing-aware scale correction (RSC) incorporates routing weights into the correction objective to mitigate scale mismatch and distortion in expert outputs introduced by ULB VQ. Our contributions are summarized as follows.

- **Structure-adaptive mixed VQ allocation.** We propose SAP, which selects an appropriate allocation signal according to the MoE output topology, and MVQ, which uses the selected signal to assign different VQ bit-widths to individual experts. This enables the limited bit budget to be allocated effectively for each MoE output topology.
- **Routing-aware output correction.** We propose RSC, which incorporates routing weights into the correction objective to account for how expert quantization errors affect the MoE block output. RSC mitigates the output distortion and scale mismatch introduced by ULB VQ.
- **Evaluation of accuracy and decode efficiency.** We evaluate *SAGE-VQ* on four MoE LLMs spanning three distinct output topologies. In particular, on Qwen2-MoE, *SAGE-VQ* limits the average accuracy degradation across seven zero-shot benchmarks to 1.42 percentage points (pp) relative to FP16 at an average expert bit-width of 2.25 bits. For decode efficiency, on Mixtral-8×7B, the mixed-bit VQ kernel achieves up to a 1.30× full-model decode speedup over the MoE-aware FP16 baseline at batch size 8.

2 Background

2.1 Mixture-of-Experts Language Models

MoE models replace the feed-forward network (FFN) in a Transformer block with multiple expert networks and a routing module (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Because the router selects only a subset of experts for each token, MoE models can increase the overall parameter capacity while limiting the amount of active computation. Given the hidden state x_t of token t, the router determines a set of top-k experts $\mathcal{E}_t$ and the corresponding routing weights $g_{t,e}$. The routed expert output is computed as

$$y_t^{\mathrm{rt}} = \sum_{e \in \mathcal{E}_t} g_{t,e} f_e(x_t), \tag{1}$$

where $f_e(\cdot)$ denotes the feed-forward transformation of expert e. Thus, expert outputs are not used independently; instead, they are weighted by token-specific routing weights and aggregated into the MoE block output.

Recent MoE architectures also differ in how they incorporate shared experts in addition to routed experts. A general MoE block output can be expressed as

$$y_t = \sum_{e \in \mathcal{E}_t} g_{t,e} f_e(x_t) + \alpha_t f_{\mathrm{sh}}(x_t), \tag{2}$$

where $f_{\mathrm{sh}}(\cdot)$ denotes the shared expert and α_t controls its contribution. In this work, we categorize MoE output topologies into three types according to the configuration of shared experts and shared gates. In Qwen-family MoE models, a shared-expert gate computes α_t, producing a shared-gated structure in which the routed output is combined with a token-dependently gated shared output (Yang et al., 2024). In DeepSeek-family models without a shared gate, the shared expert output is added directly, corresponding to $\alpha_t = 1$; we refer to this structure as shared-ungated (Liu et al., 2024a). Routed-only MoE models such as Mixtral do not contain a shared-expert path, and their MoE output is composed solely of routed expert outputs (Jiang et al., 2024). These differences in output topology change how routed expert outputs contribute to the final MoE block output.

2.2 Vector Quantization for Ultra-Low-Bit Compression

PTQ reduces the memory footprint of pretrained LLMs without costly retraining (Frantar et al., 2023; Lin et al., 2024; Xiao et al., 2023). Most PTQ methods use SQ, which maps individual weights or groups of weights to a low-precision scalar grid. In general, b-bit SQ represents each weight using one of 2^b scalar values; its limited representational capacity below 3 bits can lead to large reconstruction errors (Liu et al., 2024b).

In contrast, VQ groups multiple weights into a vector and represents each vector by a codeword index, enabling more flexible weight representation than SQ under the same bit budget. GPTVQ shows that increasing the vector dimensionality enables a more flexible quantization grid that represents the weight distribution more accurately under the same bit budget (Van Baalen et al., 2024). Recent VQ methods for dense LLMs, including additive quantization, lattice-codebook quantization, and trellis-coded quantization, have also shown effective accuracy–compression trade-offs in the ULB regime (Egiazarian et al., 2024; Tseng et al., 2024a;b). Building on these ULB VQ techniques, this work focuses on expert-wise bit allocation and quantized expert output correction that account for the structural characteristics of MoE experts.

2.3 MoE-aware Quantization

Because experts differ in their input distributions and usage frequencies, their quantization errors can have different effects on model outputs. Accordingly, recent MoE-aware quantization methods have explored mixed-precision quantization that assigns different precisions according to expert importance. MxMoE (Duanmu et al., 2025) searches for mixed-precision configurations by considering expert importance and expert activation characteristics and combines them with hardware-aware execution. AlphaQ (Yang et al., 2026a) uses the heavy-tailedness of the expert weight spectrum as a calibration-free importance metric and assigns expert-wise precision under a global bit budget. GEMQ (Deng et al., 2026) estimates model-wide importance from expert quantization error and performs global expert-level bit allocation. These methods mitigate the limitations of uniform quantization by accounting for expert importance, but each relies on a specific importance signal for bit allocation and does not explicitly consider which signal best reflects expert importance under different MoE output topologies.

Under MoE ULB quantization, expert-local errors can affect the MoE block output differently because expert outputs are weighted by routing weights before aggregation. MoEQuant (Chen et al., 2025) incorporates token–expert affinity into the quantization objective to adjust the importance of sample-wise quantization errors, while GEMQ (Deng et al., 2026) performs router adaptation to mitigate changes in token-to-expert assignments caused by low-bit quantization. More recently, KBVQ-MoE (Xu et al., 2026) addresses output distribution shift caused by accumulated quantization bias through channel-wise output correction. However, prior work has not sufficiently explored output correction objectives that explicitly account for how strongly expert quantization errors are reflected in the MoE block output according to routing weights.

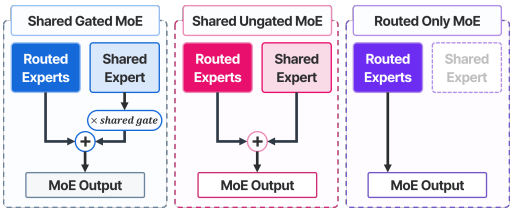

Figure 1: MoE output topologies. Shared-gated, shared-ungated, and routed-only MoE structures.

Table 1: WikiText-2 perplexity under different expert-allocation signals using the same VQ backend and average bit budget.

Model	Routing Freq.	Weight Error	Output Loss
Qwen2-MoE	**7.70**	8.57	8.11
DeepSeek-V2-Lite	9.48	12.76	**8.37**
Mixtral-8×7B	9.21	**7.61**	9.68

3 MOTIVATION

Observation 1: The Expert Allocation Signal Depends on MoE Output Topology.

Expert-wise mixed-bit allocation requires an allocation signal that determines which experts should receive higher precision so that a limited bit budget can be distributed effectively. As illustrated in Figure 1, MoE models exhibit different output topologies, which change how each expert's output is reflected in the final MoE block output. Therefore, a single allocation signal may not characterize expert importance equally well across all architectures.

To examine this, we compare perplexity (PPL) under the same VQ backend and average bit budget across three candidate allocation signals: routing frequency, weight reconstruction error, and output reconstruction loss. As summarized in Table 1, the allocation signal that yields the lowest PPL differs across models with different output topologies. Routing frequency achieves the lowest PPL on the shared-gated Qwen2-MoE, whereas output reconstruction loss is most effective on the shared-ungated DeepSeek-V2-Lite, and weight reconstruction error yields the lowest PPL on the routed-only Mixtral-8×7B. These results indicate that a single fixed allocation signal can be suboptimal across diverse MoE architectures and suggest that output topology should be considered when selecting the allocation signal.

Observation 2: Routing Weights Determine How Expert Quantization Error Affects the MoE Output.

In an MoE block, the output error introduced by expert quantization depends not only on expert-local output errors but also on the routing weights assigned to individual token–expert pairs. The resulting block output error can be written as

$$\Delta y_t = \sum_{e \in \mathcal{E}_t} g_{t,e} \left(\hat{f}_e(x_t) - f_e(x_t) \right), \quad (3)$$

where $\hat{f}_e(\cdot)$ denotes the quantized expert. As shown in Eq. 3, even local output errors of the same magnitude can have different effects on the final MoE block output depending on the routing weight assigned to each token–expert pair.

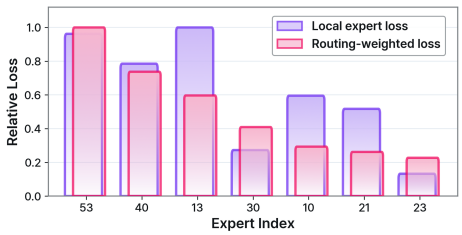

Figure 2: Comparison between local expert reconstruction error and routing-weighted output error in DeepSeek-V2-Lite Layer 18.

Figure 2 illustrates that a mismatch can arise between local expert reconstruction loss and the output error that accounts for routing weights. To compare their relative trends, we normalize the losses of the experts shown in the figure separately. An expert with a large local loss may contribute little to the final block output if its routing weights are small, whereas an expert with a smaller local loss can contribute more when it is consistently assigned larger routing weights. These observations suggest that output correction should account for routing weights in addition to expert-local errors.

Takeaway. These two observations motivate the two key designs of *SAGE-VQ*. First, SAP selects an appropriate allocation signal according to the MoE output topology, and MVQ uses the selected signal to assign VQ bit-widths to individual experts. Second, RSC corrects quantized expert outputs

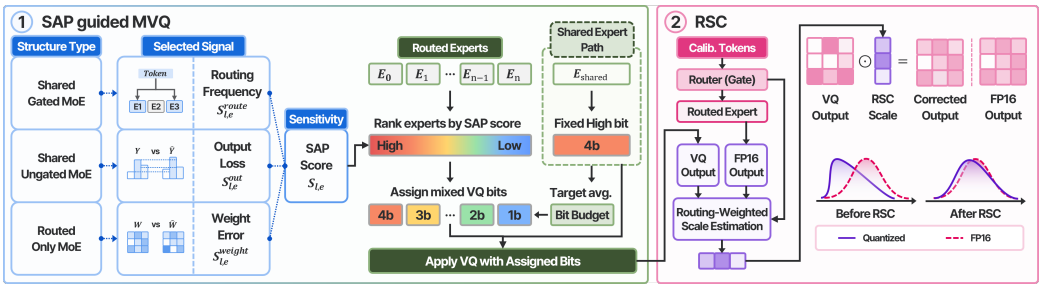

Figure 3: Overview of *SAGE-VQ*. SAP selects an allocation signal based on the MoE output topology. MVQ assigns expert-wise bit-widths through iterative priority updates within each layer under a target average bit budget, followed by VQ. RSC estimates routing-aware channel-wise correction factors to align quantized and FP16 expert outputs.

by accounting for how strongly expert output errors are reflected in the MoE block output according to the routing weights.

4 Proposed Method

4.1 Overview of SAGE-VQ

Figure 3 illustrates the overall workflow of *SAGE-VQ*. Given an FP16 MoE model and a calibration set, we collect the routing frequency, weight reconstruction error, and output reconstruction loss for each expert. SAP selects an allocation signal according to the MoE output topology, and MVQ determines the VQ bit-width of each routed expert under a target average bit budget using the selected score and then quantizes the experts accordingly. RSC subsequently corrects the scale mismatch between FP16 and quantized expert outputs using a correction objective that incorporates routing weights. The complete calibration and compression procedure is provided in Appendix A.

4.2 Structure-Adaptive Mixed Vector Quantization

Structure-adaptive allocation signal selection. Routing frequency, weight reconstruction error, and output reconstruction loss capture expert importance and quantization effects from different perspectives, and the effectiveness of each signal can vary with the MoE output topology. To account for this, SAP selects the allocation signal used for expert-wise bit allocation according to the routed/shared output structure of the target MoE.

We define three candidate allocation signals as follows. First, the routing frequency score measures how often expert e in layer l is selected by the router over the calibration token set $\mathcal{T}_l$:

$$s_{l,e}^{\text{route}} = \frac{1}{|\mathcal{T}_l|} \sum_{t \in \mathcal{T}_l} \mathbb{I}\left[e \in \mathcal{E}_{l,t}\right], \tag{4}$$

where $\mathcal{E}_{l,t}$ denotes the set of top-k experts selected for token t at layer l.

Second, weight reconstruction error measures the relative squared error introduced by probe-bit VQ. Given the expert weight $W_{l,e}$ and its probe-quantized counterpart $\widehat{W}_{l,e}^{(b_{\text{p}})}$, we define the score as

$$s_{l,e}^{\text{weight}} = \frac{\left\| W_{l,e} - \widehat{W}_{l,e}^{(b_{\text{p}})} \right\|_F^2}{\|W_{l,e}\|_F^2}. \tag{5}$$

This score normalizes the reconstruction error by the squared Frobenius norm of the original weight and is independent of calibration routing.

Third, the down-projection output loss measures reconstruction error on the activations actually routed to each expert. Let $Y_{l,e}$ and $\widehat{Y}_{l,e}^{(b_\mathrm{p})}$ denote the FP16 and probe-bit quantized down-projection outputs, respectively, for tokens routed to expert (l, e). We define the score as

$$s_{l,e}^{\mathrm{out}} = \frac{1}{n_{l,e} d_{\mathrm{out}}} \left\| Y_{l,e} - \widehat{Y}_{l,e}^{(b_\mathrm{p})} \right\|_F^2, \tag{6}$$

where $n_{l,e}$ is the number of calibration tokens routed to expert (l, e), and d_{out} is the output dimension. Unlike the weight reconstruction error, this score reflects both the expert weights and the distribution of routed activations.

SAP maps these candidate scores to MoE structures as follows:

$$s_{l,e} = \begin{cases} s_{l,e}^{\mathrm{route}}, & \text{for shared-gated MoE,} \\ s_{l,e}^{\mathrm{out}}, & \text{for shared-ungated MoE,} \\ s_{l,e}^{\mathrm{weight}}, & \text{for routed-only MoE.} \end{cases} \tag{7}$$

This topology-conditioned policy avoids applying a single fixed allocation signal uniformly across structurally different MoE models. The selected score $s_{l,e}$ is then used to determine the mixed-bit VQ bit-width of each routed expert.

Expert-wise mixed-bit VQ allocation. Once the allocation score $s_{l,e}$ is determined by SAP, MVQ assigns VQ bit-widths to the routed experts according to this score. Unlike uniform VQ, which applies the same bit-width to all routed experts, MVQ prioritizes higher precision for experts with larger scores under the selected allocation signal. We define the candidate bit-width set as $\mathcal{B} = \{1, 2, 3, 4\}$ and denote the bit-width assigned to routed expert (l, e) as $b_{l,e} \in \mathcal{B}$. The target average bit-width $\bar{b}$ is the parameter-weighted average over routed and shared expert weights. Algorithm 1 summarizes the overall allocation procedure.

Algorithm 1 Mixed Vector Quantization

Require: Expert scores $\{s_{l,e}\}$, target average bit-width $\bar{b}$, routed and shared expert sets $\mathcal{E}_\mathrm{r}$ and $\mathcal{E}_\mathrm{s}$, shared-expert precision $b_\mathrm{s} = 4$
Ensure: Expert bit map $\mathbf{B}$ and quantized routed expert weights $\{\widehat{W}_{l,e}\}$
1: Fix shared experts to b_s if present
2: Compute integer routed-expert budgets $\{C_l\}$ from $\bar{b}$, accounting for shared-expert parameter sizes and precision
3: **for** each MoE layer l **do**
4: Initialize $b_{l,e} \leftarrow 1$ for all $e \in \mathcal{E}_\mathrm{r}^l$
5: **while** $\sum_{e \in \mathcal{E}_\mathrm{r}^l} b_{l,e} < C_l$ **do**
6: $e^\star \leftarrow \arg\max_{e \in \mathcal{E}_\mathrm{r}^l : b_{l,e} < 4} \frac{s_{l,e}}{b_{l,e} + 1}$
7: $b_{l,e^\star} \leftarrow b_{l,e^\star} + 1$
8: **end while**
9: **end for**
10: **for** each routed expert (l, e) **do**
11: $\widehat{W}_{l,e} \leftarrow \mathrm{VQ}(W_{l,e}; b_{l,e})$
12: **end for**
13: **return** $\mathbf{B}$ and $\{\widehat{W}_{l,e}\}$

As shown in Algorithm 1, shared experts, when present, are fixed at 4 bits (Line 1). Their parameter sizes and fixed precision are accounted for when computing each layer's routed-expert budget C_l, expressed as the sum of its routed-expert bit-widths (Line 2). We floor the layer budgets and distribute any remaining units, up to the rounded total routed budget, by descending fractional part, breaking ties by layer index. Within each MoE layer, all routed experts are then initialized to 1 bit (Lines 3–4).

The main allocation step proceeds iteratively within each layer (Lines 5–8). At each iteration, MVQ selects the expert with the largest priority $s_{l,e}/(b_{l,e} + 1)$ among those below 4 bits and increases its bit-width by one. The selected expert's priority is updated using its new bit-width, while its allocation score remains fixed. Ties are broken by choosing the smaller expert index. Allocation terminates once the sum of routed-expert bit-widths reaches C_l.

After allocation, each routed expert is quantized at its assigned bit-width (Lines 10–12). We follow the rotation-based preprocessing of QuIP# (Tseng et al., 2024a) and apply lattice-based VQ to the rotated expert weights. Finally, MVQ returns the expert-wise bit map and the quantized routed expert weights (Line 13).

4.3 ROUTING-AWARE SCALE CORRECTION

Even after applying MVQ, ULB VQ can introduce norm and scale mismatches between FP16 and quantized expert outputs. Because routed expert outputs are weighted by token-specific routing weights before being reflected in the MoE block output, expert-local output errors of the same magnitude can have different effects on the final output depending on their routing weights. To account for this, *SAGE-VQ* applies RSC, which combines channel-wise norm correction with a least-squares correction that incorporates routing weights.

First, RSC aligns the channel-wise magnitude of each quantized projection weight with that of its corresponding FP16 weight. For projection $p \in \{\mathrm{up}, \mathrm{gate}, \mathrm{down}\}$, let W^p and $\widehat{W}^p$ denote the FP16 and quantized weights, respectively. The channel-wise norm correction factor is defined as

$$\alpha_c^{\mathrm{norm},p} = \frac{\left\|W^p_{c,:}\right\|_2}{\left\|\widehat{W}^p_{c,:}\right\|_2 + \epsilon}, \tag{8}$$

where ϵ is a small constant for numerical stability. This correction aligns the channel-wise magnitudes of the projection weights, while token-specific importance induced by routing is incorporated in the subsequent output correction.

Second, RSC estimates output-channel scales by minimizing expert output reconstruction error weighted by routing weights. For the i-th calibration token routed to an expert, let Y_i and $\widehat{Y}_i$ denote the FP16 and quantized outputs, respectively, and let ω_i denote the corresponding routing weight. RSC solves the following optimization problem:

$$\alpha^{\mathrm{ls}\star} = \arg\min_{\alpha} \sum_{i=1}^{n} \omega_i \left\| Y_i - \widehat{Y}_i \,\mathrm{diag}(\alpha) \right\|_2^2. \tag{9}$$

Thus, output errors from token–expert pairs with larger routing weights contribute more strongly to the correction objective. The closed-form solution for output channel c is

$$\alpha_c^{\mathrm{ls}\star} = \frac{\sum_{i=1}^{n} \omega_i Y_{i,c} \widehat{Y}_{i,c}}{\sum_{i=1}^{n} \omega_i \widehat{Y}_{i,c}^2 + \epsilon}. \tag{10}$$

The norm correction is applied to each projection, whereas the routing-aware least-squares correction is applied to the down projection. Both are folded into the quantized weights, requiring no additional runtime computation.

5 EXPERIMENTAL RESULTS

5.1 IMPLEMENTATION DETAILS

We evaluate *SAGE-VQ* on four representative MoE LLMs: Qwen1.5-MoE (Bai et al., 2023), Qwen2-MoE (Yang et al., 2024), DeepSeek-V2-Lite (Liu et al., 2024a), and Mixtral-8×7B (Jiang et al., 2024).

All experiments are conducted under a PTQ setting. Following common PTQ practice, we use 256 unlabeled calibration samples with a sequence length of 4096 (Frantar et al., 2023; Lin et al., 2024). In our *SAGE-VQ* experiments, we quantize MoE expert weights while keeping non-expert parameters in FP16, except for the first dense FFN of DeepSeek-V2-Lite, which is quantized to 4 bits. For models with shared experts, we fix the shared experts at 4 bits and allocate the remaining bit budget to the routed experts such that the average bit-width over all expert weights satisfies the target bit budget.

We report WikiText-2 PPL (Merity et al., 2017) and zero-shot downstream accuracy measured using lm-evaluation-harness (Gao et al., 2021). The downstream evaluation includes ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LAMBADA-OpenAI and

Table 2: Main results under ULB expert compression. W/A denotes average weight/activation precision; MxMoE and *SAGE-VQ* are evaluated at 2.25 bits, while RTN and Uniform VQ are 2-bit baselines. Wiki denotes WikiText-2 perplexity, and Avg. Acc. denotes the mean zero-shot accuracy (%) across seven downstream tasks.

W/A	Method	Qwen1.5-MoE		Qwen2-MoE		DeepSeek-V2-Lite		Mixtral-8×7B	
		Wiki ↓	Avg. Acc. ↑	Wiki ↓	Avg. Acc. ↑	Wiki ↓	Avg. Acc. ↑	Wiki ↓	Avg. Acc. ↑
16/16	FP16	6.79	68.07	5.84	72.56	5.92	70.68	3.88	78.57
2/16	RTN	57.13	31.94	445454.00	26.20	15.42	46.71	41.82	30.13
2/16	Uniform VQ	59.28	35.41	10.30	65.18	9.03	61.69	7.61	56.56
2.25/16	MxMoE	8.79	55.39	7.57	64.57	**7.01**	62.04	5.63	67.89
2.25/16	*SAGE-VQ*	**7.58**	**65.56**	**6.36**	**71.14**	**7.01**	**65.87**	**5.53**	**71.12**

LAMBADA-Standard (Paperno et al., 2016), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021); their mean accuracy is reported as Avg. We compare *SAGE-VQ* against FP16, round-to-nearest (RTN), Uniform VQ, MxMoE (Duanmu et al., 2025), and KBVQ-MoE (Xu et al., 2026). RTN is an SQ baseline with a fixed bit-width, while Uniform VQ uses the same E8-based VQ backend but applies the same bit-width to all routed experts. MxMoE serves as a strong MoE-aware mixed-precision quantization baseline.

5.2 Main Results

Table 2 compares *SAGE-VQ* with existing quantization baselines under ULB expert compression. *SAGE-VQ* achieves the highest average zero-shot accuracy among quantized methods on all four MoE LLMs and the lowest WikiText-2 PPL among quantized methods on three models, tying for the lowest PPL on DeepSeek-V2-Lite. Compared with MxMoE, a MoE-aware mixed-precision baseline, *SAGE-VQ* improves the average accuracy by 3.23–10.17 pp, demonstrating consistent performance gains across different MoE architectures.

The improvement is particularly pronounced on Qwen2-MoE. Under 2.25-bit expert compression, *SAGE-VQ* reduces WikiText-2 PPL from 7.57 to 6.36 and increases the average accuracy from 64.57 to 71.14 compared with MxMoE. Relative to FP16, the average accuracy degradation is limited to 1.42 pp, showing a strong accuracy–compression trade-off even under a highly constrained bit budget. Task-level results for the 2.25-bit comparison are provided in Appendix B.1.

Table 3: Comparison of SAGE-VQ and KBVQ-MoE in terms of WikiText-2 perplexity and average zero-shot accuracy at the 2-bit setting on two representative shared-expert MoE models.

Bit	Method	Qwen1.5-MoE		DeepSeek-V2-Lite	
		Wiki ↓	Avg. ↑	Wiki ↓	Avg. ↑
16	FP16	6.79	68.07	5.92	70.68
2	KBVQ-MoE	9.61	62.78	7.94	63.10
2	*SAGE-VQ*	**7.94**	**65.32**	**7.68**	**64.14**

Table 3 presents an additional comparison with KBVQ-MoE under the more stringent 2-bit setting. *SAGE-VQ* outperforms KBVQ-MoE in both PPL and average accuracy on Qwen1.5-MoE and DeepSeek-V2-Lite. These results show that *SAGE-VQ*, with structure-adaptive allocation and output correction, maintains strong performance even under a lower bit budget. Task-level results for this 2-bit comparison are provided in Appendix B.2.

5.3 Ablation Study

Effect of SAP. Table 4 evaluates different allocation signals at 2 bits without RSC, while Random allocation reports the mean and standard deviation over three runs. The most effective allocation signal varies with the MoE output topology. On the shared-gated Qwen2-MoE, routing frequency achieves the lowest PPL of 7.70. On the shared-ungated DeepSeek-V2-Lite, output reconstruction loss yields the lowest PPL of 8.37, outperforming routing frequency and weight reconstruction error, which result in PPLs of 9.48 and 12.76, respectively. In contrast, on the routed-only Mixtral-8×7B,

Table 4: Effect of allocation signals on WikiText-2 PPL at 2 bits. Results include MVQ only, without RSC.

Model	Random ±std.	Routing freq.	Weight error	Output loss
Qwen2-MoE	8.99 ± 0.20	**7.70**	8.57	8.11
DeepSeek-V2-Lite	15.52 ± 0.87	9.48	12.76	**8.37**
Mixtral-8×7B	47.77 ± 53.37	9.21	**7.61**	9.68

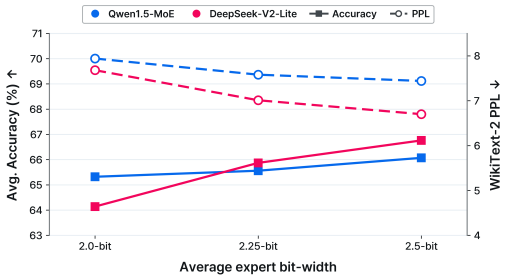

Figure 4: Bit-width–performance trade-off on Qwen1.5-MoE and DeepSeek-V2-Lite in terms of PPL and average zero-shot accuracy.

Table 5: Effect of output correction on WikiText-2 PPL at 2.25 bits. All settings use the same SAP-selected mixed-bit allocation.

Model	No correction	Plain scale	RSC
Qwen2-MoE	6.86	6.83	**6.36**
DeepSeek-V2-Lite	8.29	7.35	**7.01**
Mixtral-8×7B	6.48	6.33	**5.53**

Table 6: Full-model decode latency (ms/step) at batch size 8 on Mixtral-8×7B using two A100 80GB GPUs. Results are medians over 30 runs. P denotes initial prompt length.

Method	$P = 512$	$P = 2048$
FP16	113.77	119.47
SAGE-VQ	87.81	93.68
Speedup ↑	1.30×	1.28×

weight reconstruction error achieves the lowest PPL of 7.61. These results indicate that a single allocation signal is unlikely to be optimal across different MoE architectures. Random allocation shows high variability on Mixtral (47.77 ± 53.37); see Appendix B.3 for per-seed results and discussion.

Effect of RSC. Table 5 evaluates the effect of applying RSC to the same mixed-bit allocation selected by SAP. Plain scale corrects the scale of expert outputs without considering routing weights, whereas RSC additionally incorporates routing weights into the correction objective. Applying plain scale alone consistently reduces PPL across the evaluated models, and RSC provides further performance improvements. These results suggest that, beyond correcting scale mismatch in expert outputs under ULB quantization, incorporating routing weights into the correction objective can yield additional performance gains.

5.4 Bit-width–Performance Trade-off and Decode Efficiency

Figure 4 shows the performance–compression trade-off under 2.0-, 2.25-, and 2.5-bit settings on Qwen1.5-MoE and DeepSeek-V2-Lite. For both models, PPL and average accuracy consistently improve as the bit budget increases.

To evaluate execution efficiency, we measure full-model autoregressive decode latency at batch size 8 on Mixtral-8×7B. The FP16 baseline uses the same MoE execution framework as *SAGE-VQ*, with FP16 expert kernels, while *SAGE-VQ* uses mixed-bit VQ kernels at an average expert bit-width of 2.25 bits. As shown in Table 6, *SAGE-VQ* achieves speedups of 1.30× and 1.28× over FP16 at prompt lengths of 512 and 2048, respectively, while maintaining lower full-model decode latency than the FP16 baseline even when the overhead of compressed-weight reconstruction is included.

6 Conclusion

In this work, we proposed *SAGE-VQ*, a ULB VQ framework that combines expert-wise mixed-bit allocation tailored to MoE structures with routing-aware output correction. Experiments on four MoE LLMs showed improved accuracy–compression trade-offs over existing MoE-aware quantization methods. Additional bit-width analysis further showed that average accuracy consistently improved as the bit budget increased, indicating that *SAGE-VQ* can effectively adjust performance across different compression levels.

AI USE STATEMENT

Generative AI tools were used for English translation and language editing, feedback on experimental presentation, and assistance with code development and debugging. The core research ideas, methodology, experimental design, and conclusions were developed and validated by the authors. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

The proposed method and bit-allocation algorithm are described in Section 4, and the calibration and evaluation settings are provided in Section 5. The complete compression procedure is presented in Appendix A.

ETHICS STATEMENT

This study uses existing pretrained models and benchmark datasets and does not involve human-subject experiments or the collection of new personal or sensitive data. We have not identified additional ethical concerns specific to the proposed quantization method.

REFERENCES

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.

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*, pp. 7432–7439, 2020.

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.

Zhixuan Chen, Xing Hu, Dawei Yang, Zukang Xu, XUCHEN, Zhihang Yuan, Sifan Zhou, and JiangyongYu. MoEQuant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0epuNvt5Dj.

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.

Jianing Deng, Song Wang, Dongwei Wang, Zijie Liu, Tianlong Chen, Huanrui Yang, and Jingtong Hu. GEMQ: Global expert-level mixed-precision quantization for moe LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=5Q4ts0pr9T.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pXoZLGMNDm.

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*, pp. 12284–12303, 2024.

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. In *The Eleventh International Conference on Learning Representations*. OpenReview, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

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.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

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.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

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, 2024b.

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

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

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.

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. *arXiv preprint arXiv:1701.06538*, 2017.

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.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De. Qtip: Quantization with trellises and incoherence processing. *Advances in Neural Information Processing Systems*, 37:59597–59620, 2024b.

Mart Van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*, 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 *Proceedings of the 40th International Conference on Machine Learning*, pp. 38087–38099, 2023.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=veFs5UfYq9.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. Moe-infinity: Efficient moe inference on personal machines with sparsity-aware expert cache. *arXiv preprint arXiv:2401.14361*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *eprint arXiv: 2407.10671*, 2024.

Wanqi Yang, Yuexiao Ma, Alexander Conzelmann, Xiawu Zheng, Michael W. Mahoney, T. Konstantin Rusch, and Shiwei Liu. Alphaq: Calibration-free bit allocation for mixture-of-experts quantization. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2026a. URL https://openreview.net/forum?id=rbE8Pxs8vx.

Yuchen Yang, Yaru Zhao, Pu Yang, Shaowei Wang, and Zhi-Hua Zhou. Zipmoe: Efficient on-device moe serving via lossless compression and cache-affinity scheduling. *arXiv preprint arXiv:2601.21198*, 2026b.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. Edgemoe: Empowering sparse large language models on mobile devices. *IEEE Transactions on Mobile Computing*, 2025.

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.

A SAGE-VQ COMPRESSION PROCEDURE

Algorithm 2 presents the complete calibration and compression procedure used to implement *SAGE-VQ*. In contrast to Algorithm 1, which focuses only on the mixed-bit allocation rule, this procedure connects the three components of *SAGE-VQ*. SAP selects an architecture-dependent allocation signal, MVQ converts the selected scores into an expert-wise bit map, and RSC estimates static channel-wise correction factors that are folded into the quantized weights before inference.

The SAP stage does not assume routing frequency to be a universal allocation signal for all MoE structures. During calibration, the implementation collects or loads the candidate signals required for the target MoE structure, including routing frequency, probe-bit weight reconstruction error, and probe-bit down-projection output loss. SAP then selects one signal according to the architecture-specific rule described in Section 4.2. For shared-gated MoE models, routing frequency is used because the contribution of routed experts varies across tokens under the shared gate. For shared-ungated MoE models, output reconstruction loss is used because the shared path is always added to the routed output. For routed-only MoE models, weight reconstruction error is used because there is no shared path to compensate for distortion in the routed experts.

The MVQ stage assigns routed-expert bit-widths using the scores selected by SAP and the layer-wise allocation rule in Algorithm 1. Each routed expert is then compressed at its assigned bit-width using the QuIP# (Tseng et al., 2024a)-style VQ backend. Because the primary contribution of *SAGE-VQ* lies in its MoE-aware calibration, allocation, and correction procedure rather than in a new VQ codebook itself, the algorithm intentionally abstracts away backend-specific codebook details.

The RSC stage estimates two types of channel-wise correction factors from the calibration data. First, a projection-wise norm correction factor is computed independently for each gate, up, and down projection along its corresponding output channels. Second, a routing-weighted least-squares fit between FP16 and quantized down-projection outputs estimates a scale for each expert output channel. Because this output dimension corresponds to that of the down projection, the least-squares scale is folded only into the down-projection weights, while the projection-wise norm corrections are applied independently to their respective quantized weights. The deployed model therefore directly uses the corrected quantized weights without requiring an additional runtime correction module.

Algorithm 2 SAGE-VQ Compression Procedure

Require: FP16 MoE model M, calibration set $\mathcal{D}_{\mathrm{cal}}$, MoE structure type τ, candidate bit-widths $\mathcal{B}$, target average bit-width $\bar{b}$, routed expert set $\mathcal{E}_{\mathrm{r}}$, optional shared expert set $\mathcal{E}_{\mathrm{s}}$, shared-expert precision b_{s}

Ensure: Compressed MoE model $\widehat{M}$, expert bit map $\mathbf{B}$, and VQ/RSC metadata Ω

[SAP] Run M on $\mathcal{D}_{\mathrm{cal}}$ and collect the calibration statistics required by τ
[SAP] Build candidate allocation signals: routing frequency S^{route}, probe-bit weight error S^{weight}, and probe-bit output loss S^{out}
[SAP] Select $S \in \{S^{\mathrm{route}}, S^{\mathrm{out}}, S^{\mathrm{weight}}\}$ according to the structure rule in Section 4.2
[MVQ] Fix shared experts to b_{s} if present
[MVQ] Assign routed-expert bit-widths using the layer-wise allocation rule in Algorithm 1 under target average bit-width $\bar{b}$
for each expert projection $W^{p}_{l,e}$, $p \in \{\mathrm{gate}, \mathrm{up}, \mathrm{down}\}$ **do**
    [MVQ] Quantize $W^{p}_{l,e}$ with QuIP#-style VQ using assigned bit-width $b_{l,e}$
    [RSC] Compute per-channel norm correction $\alpha^{\mathrm{norm},p}_{l,e}$ from $W^{p}_{l,e}$ and $\widehat{W}^{p}_{l,e}$
end for
[RSC] Using FP16 down-projection inputs and norm-corrected quantized down weights, accumulate routing-weighted $\sum \omega Y \widehat{Y}$ and $\sum \omega \widehat{Y}^2$
[RSC] Compute per-channel output calibration scale $\alpha^{\mathrm{ls}\star}_{l,e} = \sum \omega Y \widehat{Y} / (\sum \omega \widehat{Y}^2 + \epsilon)$
[RSC] Fold each $\alpha^{\mathrm{norm},p}_{l,e}$ into its corresponding quantized projection $\widehat{W}^{p}_{l,e}$
[RSC] Fold $\alpha^{\mathrm{ls}\star}_{l,e}$ into the quantized down-projection weights $\widehat{W}^{\mathrm{down}}_{l,e}$
Store the bit map, VQ entries, and fused correction metadata in Ω
return $\widehat{M}$, $\mathbf{B}$, and Ω

B Detailed Results

This section reports task-level scores corresponding to the aggregate results in Section 5.2. These results provide a more detailed view of how each compression method affects the individual downstream benchmarks that contribute to the average accuracy. In our *SAGE-VQ* experiments, activations remain in FP16, and MoE expert weights are quantized. Non-expert parameters remain in FP16, except for the first dense FFN of DeepSeek-V2-Lite, which is quantized to 4 bits. Wiki denotes WikiText-2 PPL, where lower is better. Avg. denotes the average zero-shot accuracy across seven downstream benchmarks, where higher is better.

The downstream columns consist of ARC-c, ARC-e, HellaSwag, LAMBADA-O, LAMBADA-S, PIQA, and WinoGrande, following the lm-evaluation-harness protocol used in Section 5. These tasks evaluate different aspects of language-model behavior, including commonsense reasoning, sentence completion, and physical reasoning.

B.1 Detailed 2.25-bit Results Compared with MxMoE

Tables 7–10 provide task-level results for the 2.25-bit comparison summarized in Table 2. FP16 denotes the uncompressed baseline. RTN and Uniform VQ are internal baselines under aggressive compression and are evaluated at 2-bit, whereas MxMoE and *SAGE-VQ* are evaluated at 2.25 bits to compare mixed-precision expert compression under the same average expert-weight budget.

Table 7 reports the results on Qwen1.5-MoE. The results of RTN and Uniform VQ show that direct low-bit compression can substantially degrade zero-shot accuracy, particularly on LAMBADA and reasoning-oriented benchmarks. MxMoE recovers part of this degradation through MoE-aware mixed precision, but a substantial gap from FP16 remains. *SAGE-VQ* considerably narrows this gap, increasing the average accuracy by 10.17 pp over MxMoE while reducing PPL from 8.79 to 7.58. These results suggest that structure-adaptive VQ allocation, which selects the allocation signal according to the MoE topology, can effectively preserve overall downstream performance.

Table 8 reports results on Qwen2-MoE, the shared-gated model emphasized in the main text. This model combines routed experts with a gated shared-expert path, making it particularly relevant because both the allocation signal and output correction are sensitive to routing behavior. Against MxMoE, *SAGE-VQ* improves on every reported task and reduces PPL by 1.21. Its final average accuracy reaches 71.14, within 1.42 pp of the FP16 baseline, showing that the proposed SAP-based MVQ and RSC can preserve near-FP16 behavior even under 2.25-bit expert compression.

Table 9 evaluates DeepSeek-V2-Lite, a shared-ungated MoE model. In this setting, *SAGE-VQ* ties MxMoE in WikiText-2 PPL while increasing the average zero-shot accuracy from 62.04 to 65.87. This indicates that matching PPL alone is not sufficient when evaluating MoE compression. The task-level results show that, even at the same PPL, *SAGE-VQ* better preserves downstream accuracy.

Table 10 reports the results on Mixtral-8$\times$7B, a routed-only MoE architecture without a shared-expert path. *SAGE-VQ* increases the average accuracy by 3.23 pp over MxMoE while slightly reducing PPL from 5.63 to 5.53. The largest gains appear on ARC-Challenge, ARC-Easy, PIQA, and WinoGrande. These results suggest that a structure-adaptive allocation signal remains useful even when the MoE architecture differs substantially from the shared-gated Qwen family.

B.2 Detailed 2-bit Results Compared with KBVQ-MoE

Tables 11 and 12 report comparisons with KBVQ-MoE under the more stringent 2-bit setting, where the VQ codebook capacity is smaller and compression error is more severe.

Table 11 shows that *SAGE-VQ* remains effective under 2-bit quantization on Qwen1.5-MoE, reducing PPL from 9.61 to 7.94 and increasing Avg. from 62.78 to 65.32 over KBVQ-MoE. Gains across multiple zero-shot tasks further support the benefit of expert-aware allocation and routing-aware correction under the same bit budget.

Table 12 reports the 2-bit comparison results on DeepSeek-V2-Lite. Compared with KBVQ-MoE, *SAGE-VQ* reduces PPL from 7.94 to 7.68 and increases Avg. from 63.10 to 64.14. These improvements across multiple tasks on a shared-ungated architecture support the structure-adaptive allocation of SAP.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	44.03	69.53	77.26	71.28	64.62	80.47	69.30	68.07	6.79
RTN	2/16	24.66	36.36	43.33	5.26	4.21	61.32	48.46	31.94	57.13
Uniform VQ	2/16	24.91	32.11	39.77	22.74	17.37	58.27	52.72	35.41	59.28
MxMoE	2.25/16	31.66	53.28	62.80	56.43	51.00	71.33	61.25	55.39	8.79
SAGE-VQ	2.25/16	42.24	68.06	73.69	67.63	62.27	78.84	66.22	65.56	7.58

Table 7: Detailed results on Qwen1.5-MoE. RTN and Uniform VQ are evaluated at 2 bits, while MxMoE and *SAGE-VQ* are evaluated at 2.25 bits.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	55.20	77.19	84.09	74.35	62.62	82.32	72.14	72.56	5.84
RTN	2/16	25.00	26.98	28.23	0.00	0.02	52.67	50.51	26.20	445454.00
Uniform VQ	2/16	44.88	68.52	72.68	67.42	58.96	76.55	67.25	65.18	10.30
MxMoE	2.25/16	45.05	68.86	77.13	66.00	56.61	75.41	62.90	64.57	7.57
SAGE-VQ	2.25/16	53.67	74.66	82.46	72.23	61.87	81.56	71.51	71.14	6.36

Table 8: Detailed results on Qwen2-MoE. RTN and Uniform VQ are evaluated at 2 bits, while MxMoE and *SAGE-VQ* are evaluated at 2.25 bits.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	48.98	76.22	77.91	72.33	67.90	80.20	71.19	70.68	5.92
RTN	2/16	33.87	53.66	57.19	30.93	22.90	70.08	58.33	46.71	15.42
Uniform VQ	2/16	41.04	67.17	69.59	57.56	53.13	76.22	67.09	61.69	9.03
MxMoE	2.25/16	40.36	68.86	68.63	59.56	54.01	75.08	67.80	62.04	7.01
SAGE-VQ	2.25/16	44.97	70.12	73.02	66.60	60.10	78.29	67.96	65.87	7.01

Table 9: Detailed results on DeepSeek-V2-Lite. RTN and Uniform VQ are evaluated at 2 bits, while MxMoE and *SAGE-VQ* are evaluated at 2.25 bits.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	66.38	85.39	85.95	77.28	73.06	85.20	76.72	78.57	3.88
RTN	2/16	25.43	34.34	34.91	4.66	5.08	56.58	49.88	30.13	41.82
Uniform VQ	2/16	31.83	51.43	70.81	61.03	49.31	68.93	62.59	56.56	7.61
MxMoE	2.25/16	48.98	72.77	77.44	68.68	62.18	76.28	68.90	67.89	5.63
SAGE-VQ	2.25/16	54.61	77.48	79.94	69.86	62.16	81.56	72.22	71.12	5.53

Table 10: Detailed results on Mixtral-8×7B. RTN and Uniform VQ are evaluated at 2 bits, while MxMoE and *SAGE-VQ* are evaluated at 2.25 bits.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	44.03	69.53	77.26	71.28	64.62	80.47	69.30	68.07	6.79
RTN	2/16	24.66	36.36	43.33	5.26	4.21	61.32	48.46	31.94	57.13
Uniform VQ	2/16	24.91	32.11	39.77	22.74	17.37	58.27	52.72	35.41	59.28
KBVQ-MoE	2/16	39.59	65.66	64.52	67.16	60.20	77.20	65.11	62.78	9.61
SAGE-VQ	2/16	43.52	68.90	72.17	66.70	60.53	78.02	67.40	65.32	7.94

Table 11: Detailed 2-bit results on Qwen1.5-MoE compared with KBVQ-MoE.

Method	W/A	ARC-c	ARC-e	HellaSwag	LAMBADA-O	LAMBADA-S	PIQA	WinoGrande	Avg. ↑	PPL ↓
FP16	16/16	48.98	76.22	77.91	72.33	67.90	80.20	71.19	70.68	5.92
RTN	2/16	33.87	53.66	57.19	30.93	22.90	70.08	58.33	46.71	15.42
Uniform VQ	2/16	41.04	67.17	69.59	57.56	53.13	76.22	67.09	61.69	9.03
KBVQ-MoE	2/16	40.60	68.47	67.93	64.91	55.94	75.91	67.91	63.10	7.94
SAGE-VQ	2/16	44.03	69.70	70.81	63.32	56.98	76.17	67.96	64.14	7.68

Table 12: Detailed 2-bit results on DeepSeek-V2-Lite compared with KBVQ-MoE.

Table 13: Per-seed results for Random allocation at a parameter-weighted average 2-bit expert budget without RSC. Avg. Acc. is the mean accuracy over seven zero-shot tasks. Routed expert counts are aggregated across all MoE layers.

Model	Seed	PPL ↓	Avg. Acc. (%) ↑	Routed expert counts			
				1-bit	2-bit	3-bit	4-bit
Qwen2-MoE	42	8.76	70.06	891	474	411	16
	43	9.04	69.99	906	456	402	28
	44	9.15	69.52	912	458	380	42
Mixtral-8×7B	42	20.41	45.08	114	55	60	27
	43	13.62	45.89	103	69	65	19
	44	109.27	29.61	119	47	61	29
DeepSeek-V2-Lite	42	15.22	54.77	738	409	400	117
	43	14.83	55.23	749	393	399	123
	44	16.50	49.01	773	367	379	145

B.3 Random Allocation Variability

Experimental setup. Table 13 provides per-seed results for the Random baseline in Table 4. Random allocations are generated by applying the allocation rule in Algorithm 1 to seeded random expert scores sampled uniformly from $[0, 1)$. We evaluate allocation seeds 42, 43, and 44 while fixing the model and calibration seeds to 42. All runs use E8-based VQ without RSC, with routed-expert candidate precisions of 1/2/3/4 bits and shared experts, when present, fixed at 4 bits. The parameter-weighted average bit-width across routed and shared expert weights is 2 bits. For DeepSeek-V2-Lite, the budget covers its 26 MoE layers; the first dense FFN is fixed at 4 bits. WikiText-2 PPL is evaluated at a sequence length of 4096, and average accuracy is computed over the seven zero-shot tasks used in the main evaluation.

Possible explanation. Mixtral-8×7B activates only two of its eight experts per layer for each token (Jiang et al., 2024). We hypothesize that this reliance on a small set of active experts may make token-level MoE outputs sensitive to unfavorable low-bit assignments to the selected experts, potentially contributing to the severe degradation and large variability observed under Random allocation.