BMoEQ: Binary-Coded MoE Quantization with Runtime-Adaptive Mixed-Precision Support

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts large language models (MoE-LLMs) scale model capacity through sparse activation but retain a substantial memory footprint from expert weights. Existing expert-level mixed-precision methods alleviate this cost by fixing bit-widths according to calibration-based importance estimates, which may not generalize across inputs, domains, and tasks. We propose BMoEQ, a quantization method that combines uniform ultra-low-bit base precision across experts with input-adaptive execution precision through binary-coded quantization (BCQ). To improve quantization accuracy without static expert-level mixed-precision allocation, BMoEQ incorporates saliency-aware error reconstruction, assigning an additional bit plane to a small subset of salient weights while maintaining a common base bit-width across experts. At runtime, online importance scores guide the selective omission of bit planes for less important activated experts, reducing memory traffic and computation without removing their contributions entirely. A single stored representation thus supports multiple execution precisions without requantization or separate lower-precision copies, while preserving access to each expert's full stored precision. We further develop hardware-efficient GPU kernels that selectively load and process the required bit planes and fuse the salient-weight correction path to minimize latency overhead. Experiments show that the saliency-aware quantization in BMoEQ achieves superior overall accuracy in both 2-bit and 3-bit settings, outperforming expert-level mixed-precision baselines at comparable effective bit-widths. With runtime precision adaptation and optimized GPU kernels, BMoEQ achieves up to 6.4× kernel speedup over the FP16 cuBLAS baseline. Code is available at `https://anonymous.4open.science/r/BMoEQ-B345`.

1 Introduction

Mixture-of-Experts large language models (MoE-LLMs) have emerged as an effective approach to scaling model capacity while limiting per-token computation (Shazeer et al., 2017; Muennighoff et al., 2025; Jiang et al., 2024; Yang et al., 2025). By activating only a small subset of experts for each input, these models increase their total parameter count without a proportional increase in computational cost. However, sparse activation does not eliminate the storage requirements of inactive experts, leaving a substantial memory footprint that complicates deployment(Duanmu et al., 2025). This has motivated ultra-low-bit expert quantization, particularly expert-level mixed-precision approaches that exploit heterogeneity in expert importance (Huang et al., 2025; Deng et al., 2026). By allocating bit-widths according to offline importance estimates, these methods can achieve higher accuracy than uniform-precision quantization under the same bit budget.

Despite their effectiveness, these approaches typically determine expert precision assignments from calibration statistics and keep them fixed throughout inference. Such assignments may not reflect changes in expert importance across inputs, domains, and tasks. For example, a configuration that preserves accuracy on tasks aligned with the calibration distribution may incur larger accuracy losses on underrepresented tasks, such as mathematical reasoning (Cobbe et al., 2021) or code generation (Chen et al., 2021). An expert deemed less important during calibration may become critical for a downstream task, yet its precision remains limited by the lower bit-width assigned offline. Consequently, static mixed-precision allocation can lead to substantial accuracy degradation in domains underrepresented during calibration, particularly under aggressive compression.

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.

This mismatch motivates an alternative approach: maintaining a uniform base precision across experts while adapting their execution precision to input-dependent importance. Realizing this approach requires addressing two challenges. First, conventional uniform-precision methods often underperform expert-level mixed-precision methods at the same effective bit-width. Second, a straightforward implementation of runtime precision adaptation can incur substantial memory overhead if separate weight representations are stored for each precision level.

To address these challenges, we propose BMoEQ, a quantization method for MoE models that combines uniform ultra-low-bit base quantization across experts with dynamic mixed-precision support. BMoEQ leverages the bit-plane structure of binary-coded quantization (BCQ) (Xu et al., 2018), which enables individual planes to be stored and processed independently. To overcome the first challenge, we incorporate saliency-aware error reconstruction (Park et al., 2026b) tailored to MoE models. Specifically, we allocate an additional bit plane to a small subset of salient weights, forming an auxiliary accumulation path that compensates for quantization errors. Because BCQ-based matrix multiplication is performed by iteratively accumulating the bit planes, the error reconstruction path using a single bit plane incurs negligible latency overhead, while improving ultra-low-bit quantization accuracy. Importantly, this refinement operates at the weight level while maintaining the same base bit-width across experts, without relying on static expert-level mixed-precision allocation.

To address the second challenge, BMoEQ leverages the same representation for runtime-adaptive quantization guided by online expert importance scores. For each input, we reduce the execution precision of activated experts estimated to contribute less by selectively omitting bit planes from their stored representation. A single representation thus supports multiple execution precisions without runtime requantization or separate lower-precision weight copies. Crucially, each expert retains access to its full stored precision and can use it when its estimated importance increases. Unlike dynamic expert pruning (Lu et al., 2024), this approach reduces the precision of selected experts rather than removing their contributions entirely. Finally, we develop hardware-efficient GPU kernels that translate runtime precision adaptation into lower memory traffic and computational cost by loading and processing only the required bit planes. We further fuse the salient-weight correction path into the BCQ kernel (Park et al., 2024) to minimize latency overhead.

Our experiments show that the saliency-aware quantization in BMoEQ achieves superior overall accuracy in both the 2-bit and 3-bit settings, outperforming expert-level mixed-precision baselines at comparable effective bit-widths. With runtime precision adaptation and optimized GPU kernels, BMoEQ achieves up to 6.4× kernel speedup over FP16 cuBLAS, demonstrating the benefits of jointly optimizing low-bit expert representations, input-adaptive precision, and GPU execution.

2 PRELIMINARIES

Mixture-of-Experts LLMs. Mixture-of-Experts (MoE) (Shazeer et al., 2017) enables efficient LLM scaling by replacing dense MLPs in Transformer blocks with sparsely activated expert networks. Each MoE layer comprises N MLP-based experts and a router that selects only the top-k experts for each input token based on routing scores. Unlike a dense MLP, this sparse execution introduces routing and aggregation: tokens are dispatched to the selected experts, and their outputs are combined through a routing-weighted sum, preserving the original token dimensionality. For each input token t, the MoE output is expressed as:

$$\boldsymbol{y}_t = \sum_{e \in \mathcal{E}_t} r_{t,e} \cdot E_e(\boldsymbol{x}_t), \qquad E_e(\boldsymbol{x}_t) = \boldsymbol{W}_D^e \left[\sigma(\boldsymbol{W}_G^e \cdot \boldsymbol{x}_t) \odot (\boldsymbol{W}_U^e \cdot \boldsymbol{x}_t)\right], \tag{1}$$

where $\mathcal{E}_t$ denotes the top-k selected experts, $r_{t,e}$ is the corresponding routing score, and $\boldsymbol{W}_U^e$, $\boldsymbol{W}_G^e$, and $\boldsymbol{W}_D^e$ denote the up-, gate-, and down-projection weights, respectively. Here, $\sigma(\cdot)$ denotes the activation function.

MoE Quantization. Quantization maps high-precision values to discrete lower-precision levels, reducing the memory footprint of LLM parameters. Conventional symmetric integer quantization with max-scaling is expressed as:

$$\boldsymbol{X}_q = \text{clip}(\lceil \boldsymbol{X}/\boldsymbol{s} \rfloor), \quad \boldsymbol{s} = \max(|\boldsymbol{X}|)/(2^{n-1}-1), \quad \hat{\boldsymbol{X}} = \boldsymbol{s} \cdot \boldsymbol{X}_q \tag{2}$$

where $\boldsymbol{s}$ denotes the scaling factor, n is the target bit-width, $\lfloor \cdot \rceil$ denotes round-to-nearest (RTN), and $\mathrm{clip}(\cdot)$ restricts the integer codes to the symmetric range $[-(2^{n-1}-1),\ 2^{n-1}-1]$. Quantization is particularly beneficial for MoE-LLMs, as expert weights dominate model parameters. MoEQuant (Hu et al., 2025) adapts prior quantization to MoE-specific characteristics through expert-balanced self-sampling and affinity-guided quantization, thereby reducing the memory footprint of expert weights.

Recent studies (Huang et al., 2025; Deng et al., 2026) further demonstrate aggressive compression to sub-4-bit precision by applying expert-level mixed-precision quantization that exploits heterogeneity across experts to assign different bit-widths. Specifically, PMQ (Huang et al., 2025) formulates expert-level bit allocation as a linear programming problem that jointly considers activation frequency, routing scores, and quantization-induced reconstruction error under a target bit budget. GEMQ (Deng et al., 2026) extends this direction by estimating expert importance globally from quantization-induced task-loss perturbations and performing model-wide expert-level bit allocation.

Binary-Coded Quantization (BCQ). BCQ represents a weight matrix as a linear combination of bit-planes and scaling factors. A q-bit BCQ weight matrix is represented as:

$$\widehat{\boldsymbol{W}} = \sum_{i=1}^{q} \boldsymbol{\alpha}_i \cdot \boldsymbol{B}_i, \quad \boldsymbol{B}_i \in \{-1, +1\}^{m \times n}. \tag{3}$$

Algorithm 1 Alternating optimization of q-bit BCQ

Input: Original weight $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, bit-width q, refinement iterations T
Output: Scaling factors $\{\boldsymbol{\alpha}_i\}_{i=1}^{q}$ and bit-planes $\{\boldsymbol{B}_i\}_{i=1}^{q}$
function ALTERNATING BCQ($\boldsymbol{W}, q, T$)
  $\{\boldsymbol{\alpha}_i, \boldsymbol{B}_i\}_{i=1}^{q} \leftarrow$ GREEDY($\boldsymbol{W}$)
  for $t = 1$ to T **do**
    $\{\boldsymbol{\alpha}_i\}_{i=1}^{q} \leftarrow \mathrm{LS}(\boldsymbol{B}, \boldsymbol{W})$
    $\{\boldsymbol{B}_i\}_{i=1}^{q} \leftarrow \mathrm{BS}(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q, \boldsymbol{W})$
  end for
end function

As shown in Alg. 1, successive bit-planes $\boldsymbol{B}$ and their scaling factors $\boldsymbol{\alpha}$ progressively refine the remaining quantization error. For LLM quantization, BCQ additionally applies block-wise reconstruction(Li et al., 2021) by updating only the scaling factors(Park et al., 2026a). This bit-plane representation enables selective memory access and computation for flexible runtime precision. For efficient hardware execution, BCQ replaces explicit dot products with LUT-based computation(Park et al., 2024): partial sums over groups of μ activations are precomputed in a LUT, and linear operations are performed through LUT lookups and accumulations instead of conventional multiply-and-accumulate operations.

3 MOTIVATION

Limitation of Existing Mixed-Precision Approaches. Recent mixed-precision studies (Huang et al., 2025; Deng et al., 2026) have shown that experts within a MoE contribute unequally to model performance, motivating higher precision for important experts and more aggressive quantization for less important ones. Existing approaches, however, typically rely on static precision assignment, where expert importance is estimated from a small general-domain calibration dataset (e.g., WikiText-2 or C4) and the resulting expert-level bit-width configuration is fixed prior to deployment.

Meanwhile, expert importance is not an intrinsic static property of the model, but varies with the input distribution. As illustrated in Fig. 5, expert importance exhibits markedly different distributions between general-domain text (e.g., WikiText-2) and mathematical reasoning tasks (e.g., GSM8K). Consequently, runtime inputs from domains such as mathematics or code generation can produce importance patterns that deviate substantially from calibration-derived estimates. As shown in Fig. 1(b), this mismatch causes greater performance degradation on domain-specific tasks, exposing a fundamental limitation of static mixed-precision quantization: fixed expert-level bit-width assignments cannot accommodate runtime changes in expert importance. These observations motivate a quantization framework that adapts expert precision at runtime to exploit the dynamic heterogeneity of MoE models, while preserving the hardware efficiency of mixed-precision execution.

BCQ for Dynamic Mixed-Precision Support. While dynamic mixed-precision support is attractive, a naïve implementation may still fetch weights at the highest supported precision, limiting the latency benefit of lower-bit execution. In contrast, BCQ organizes weights along the bit-plane dimension, allowing unused bit planes to be skipped together with their associated weight fetches, LUT lookups, and accumulations. This structure enables both memory access and computation to scale down as runtime precision decreases. BCQ also offers a favorable numerical property for runtime precision reduction. Unlike conventional integer quantization, where truncation error is tied

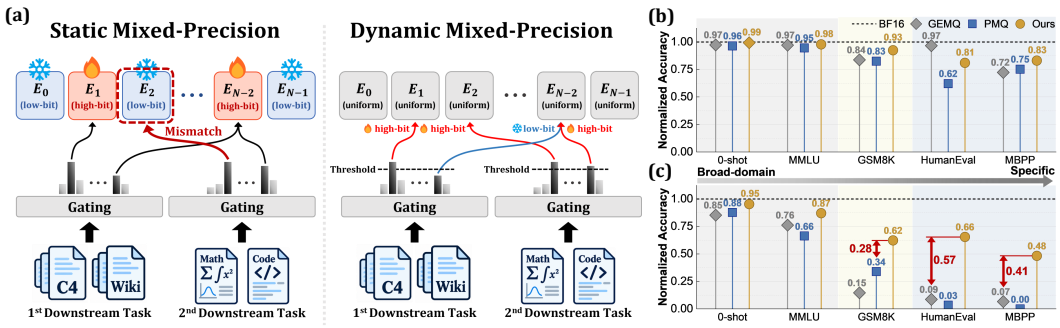

Figure 1: (a) Static mixed-precision (left) vs. dynamic mixed-precision (right) across domains. Static methods fix expert bit-widths offline, causing a mismatch with runtime importance under domain shifts. We report normalized accuracy of Mixtral-8×7B-v0.1 (b) at 3-bit and (c) 2-bit. In both settings, the gap between static methods and BMoEQ widens on domain-specific tasks.

to a quantization step derived from the original weight range, each BCQ bit plane is fitted to the residual left by preceding planes. As derived in Appendix A.2, this progressive refinement yields smaller truncation-induced perturbations, making BCQ well suited for runtime precision reduction.

Meanwhile, dynamic mixed-precision execution still faces a key limitation: the total expert storage footprint is determined by the base uniform precision retained for all experts. Despite the strong numerical advantages of BCQ in the ultra-low-bit regime, uniform BCQ still suffers greater accuracy degradation than expert-level optimized mixed-precision quantization. To address this limitation, we incorporate saliency-aware error reconstruction (Zhang et al., 2024; Park et al., 2026b) into BCQ by assigning an additional correction bit plane to a small subset of important weights. This improves quantization accuracy while preserving a uniform base precision across experts, allowing BCQ to achieve accuracy comparable to static mixed-precision methods at a similar effective bit-width. By fully exploiting the bit-plane structure of BCQ, we combine saliency-aware uniform quantization with runtime precision adaptation, enabling low-latency and memory-efficient MoE-LLM serving.

4 METHOD

We propose BMoEQ, which combines saliency-aware BCQ for accurate ultra-low-bit quantization with Runtime-Adaptive Mixed-Precision (RMP) for efficient MoE inference. Sec. 4.1 presents the proposed BCQ error reconstruction based on projection-wise joint saliency and expert-wise joint permutation. Sec. 4.2 introduces runtime importance scoring and layer-aware precision adaptation for dynamic bit-plane execution.

4.1 SALIENCY-AWARE BCQ

Joint-Saliency Estimation. As discussed in Sec. 3, while BCQ is well suited to the ultra-low-bit regime, uniform-precision BCQ still suffers greater accuracy degradation than mixed-precision approaches. We therefore incorporate error reconstruction methods (Zhang et al., 2024; Park et al., 2026b) to recover accuracy by compensating for residual quantization errors. Specifically, we apply error reconstruction directly to BCQ by assigning a single additional bit plane only to salient weights, introducing just 0.125 effective bits per weight—comparable to the overhead of integer scaling factors—with only marginal latency overhead in the fused BCQ kernel.

Fig. 2 illustrates the proposed BCQ joint-saliency estimation for expert linear layers. The goal is to minimize propagated quantization error, where saliency is measured using the quantization-induced output perturbation of each linear layer:

$$\Delta \boldsymbol{Y} = \Delta \boldsymbol{W} \cdot \boldsymbol{X}, \quad \Delta \boldsymbol{W} = \boldsymbol{W} - \mathcal{Q}(\boldsymbol{W}), \tag{4}$$

where $\mathcal{Q}(\cdot)$ denotes the quantization. We adopt a coarse-grained criterion that selects salient channels along the intermediate dimension. Since these channels may remain scattered, we cluster them into a contiguous range for regular memory access to the auxiliary bit-plane. Because this dimension is shared by the up-, gate-, and down-projections and is permutation-symmetric, the reordering

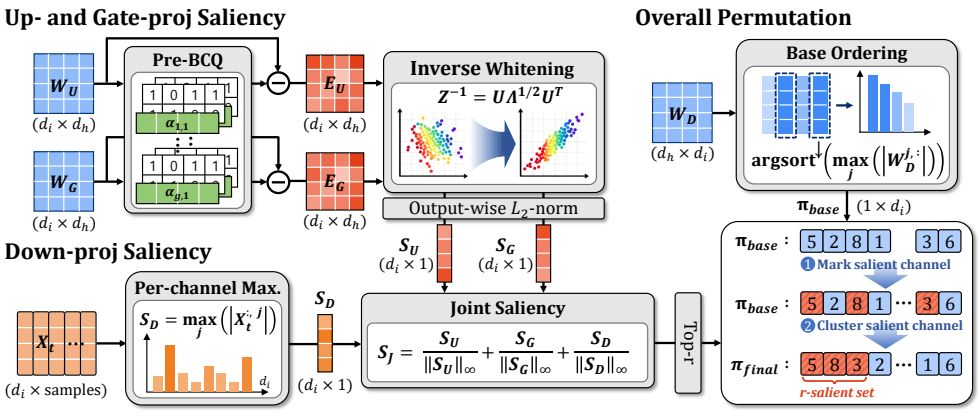

Figure 2: Overview of saliency-aware BCQ. Salient channels are identified along the shared intermediate dimension of the up-, gate-, and down-projections. Projection-wise saliency is combined to select the top-r channels, which are moved to the front to form the joint weight permutation.

is fully absorbed into the expert weights offline with no runtime overhead. For the up- and gate-projections, we measure channel saliency from the quantization error weighted by the input distribution using the inverse ZCA matrix $\boldsymbol{Z}^{-1} = \boldsymbol{U}\Lambda^{1/2}\boldsymbol{U}^\top$, derived from $\boldsymbol{H} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{U}\Lambda\boldsymbol{U}^\top$. For the down-projection, whose weights are approximately normally distributed, we instead use activation magnitude to identify intermediate channels that dominate the accumulated quantization error. The projection-wise saliency scores are defined as:

$$\boldsymbol{S}_U^i = \left\|(\Delta \boldsymbol{W}_U \cdot \boldsymbol{Z}^{-1})_{i,:}\right\|_2, \quad \boldsymbol{S}_G^i = \left\|(\Delta \boldsymbol{W}_G \cdot \boldsymbol{Z}^{-1})_{i,:}\right\|_2, \quad \boldsymbol{S}_D = \max_j \left|\boldsymbol{X}_t^{:,j}\right|, \tag{5}$$

where $\boldsymbol{S}_U, \boldsymbol{S}_G, \boldsymbol{S}_D \in \mathbb{R}^{d_i \times 1}$, and $\boldsymbol{X}_t \in \mathbb{R}^{d_i \times \text{samples}}$ denotes the collected intermediate activations. We normalize the three projection-wise saliencies and combine them into a joint weight saliency:

$$\boldsymbol{S}_J = \frac{\boldsymbol{S}_U}{\|\boldsymbol{S}_U\|_\infty} + \frac{\boldsymbol{S}_G}{\|\boldsymbol{S}_G\|_\infty} + \frac{\boldsymbol{S}_D}{\|\boldsymbol{S}_D\|_\infty} \tag{6}$$

Expert-wise Joint Weight Permutation. Given the joint saliency, we first construct a base ordering from the per-channel maximum values of the down-projection weights:

$$\boldsymbol{\pi}_{\text{base}} = \text{argsort}^{\downarrow}\left(\max_j \left|\boldsymbol{W}_D^{j,:}\right|\right), \tag{7}$$

This ordering separates weight outliers and inliers across quantization groups, improving the down-projection quantization while having no effect on the up- and gate-projections, where the same permutation only reorders output channels. We then select the top-r salient intermediate channels $\mathcal{S}_r$ from $\boldsymbol{S}_J$ and cluster them at the front while preserving the base ordering:

$$\boldsymbol{\pi}_{\text{final}} = \left[\boldsymbol{\pi}_{\text{base}}\big|_{\mathcal{S}_r} \;;\; \boldsymbol{\pi}_{\text{base}}\big|_{\mathcal{S}_r^c}\right], \qquad \boldsymbol{P}^{i,j} = \mathbf{1}[j = \boldsymbol{\pi}_{\text{final}}^i]. \tag{8}$$

The resulting permutation matrix $\boldsymbol{P}$ exploits the intermediate-channel permutation symmetry within each expert. We jointly transform the projection weights as:

$$\widetilde{\boldsymbol{W}}_U^e = \boldsymbol{P}_e \cdot \boldsymbol{W}_U^e, \qquad \widetilde{\boldsymbol{W}}_G^e = \boldsymbol{P}_e \cdot \boldsymbol{W}_G^e, \qquad \widetilde{\boldsymbol{W}}_D^e = \boldsymbol{W}_D^e \cdot \boldsymbol{P}_e^\top. \tag{9}$$

Because the element-wise activation function and multiplication commute with a common intermediate-channel permutation, the transformed expert output satisfies:

$$\widetilde{E}_e(\boldsymbol{x}) = \boldsymbol{W}_D^e \boldsymbol{P}_e^\top \cdot \boldsymbol{P}_e \left[\sigma\left(\boldsymbol{W}_G^e \cdot \boldsymbol{x}\right) \odot \left(\boldsymbol{W}_U^e \cdot \boldsymbol{x}\right)\right] = E_e(\boldsymbol{x}). \tag{10}$$

Thus, the permutation preserves mathematical invariance and is fully absorbed into the expert weights offline. Unlike hidden-dimension permutation, which requires expert-specific runtime reordering at the input or output, intermediate-channel permutation preserves the existing MoE routing

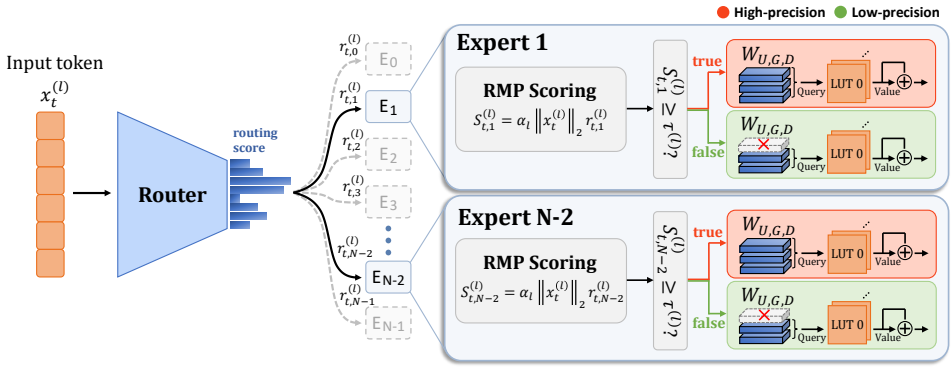

Figure 3: Inference-time execution flow of runtime-adaptive mixed-precision (RMP). During inference, RMP scores each routed expert using the routing score and input-token l_2-norm, then selects its precision through a layer-specific threshold. Lower-precision paths skip the final bit-plane together with the corresponding weight fetch, lookup, and accumulation operations.

and aggregation system. Consequently, the selected salient channels form a contiguous region across all three projections, enabling regular access to the auxiliary bit-plane without additional runtime reordering.

4.2 Runtime-Adaptive Mixed-Precision Support

Runtime Expert Importance. Prior expert-level mixed-precision methods exploit expert heterogeneity primarily to reduce model storage by assigning expert-specific bit-widths from offline calibration. As discussed in Sec. 3, static allocation cannot reflect input-dependent changes in expert importance. We instead exploit this dynamic heterogeneity during inference through Runtime-Adaptive Mixed-Precision (RMP), which adapts the precision of each routed expert for efficient MoE inference.

We define runtime expert importance according to its contribution to the MoE output defined in equation 1. Since each expert contribution is weighted by its routing score, $r_{t,e}^{(l)}$ directly reflects its contribution to the final MoE output. However, $E_e^{(l)}(\boldsymbol{x}_t^{(l)})$ is unavailable before expert computation, precisely when the precision decision must be made. We therefore use the l_2-norm of the input representation as a lightweight proxy for the expert output magnitude and define the runtime importance score as:

$$s_{t,e}^{(l)} = r_{t,e}^{(l)} \cdot \left\|\mathbf{x}_t^{(l)}\right\|_2. \tag{11}$$

This score provides a lightweight estimate of expert importance prior to expert execution. RMP then compares $s_{t,e}^{(l)}$ with the layer-specific threshold ($\tau^{(l)}$) and selects the runtime precision from the base precision (B) as:

$$b_{t,e}^{(l)} = \begin{cases} B-1, & s_{t,e}^{(l)} < \tau^{(l)}, \\ B, & s_{t,e}^{(l)} \geq \tau^{(l)}, \end{cases} \qquad \tau^{(l)} = \alpha^{(l)}\, \tau_{\text{base}}^{(l)}, \tag{12}$$

The selection of $\tau^{(l)}$ is discussed in the following paragraph. RMP selects between two execution paths: Important experts (i.e., $s_{t,e}^{(l)} \geq \tau^{(l)}$) use all (B) bit planes, whereas less important experts use only the first (B-1), skipping the final bit-plane fetch, LUT lookup, and accumulation (Fig. 3). Consequently, the fused GPU kernels equipped with RMP directly translate runtime expert importance into reduced memory traffic and computation, yielding lower MoE inference latency.

Layer-wise Threshold Adjustment. A base threshold alone does not account for differences in precision sensitivity across MoE layers. As shown in Fig. 6, deeper layers exhibit greater accuracy degradation under lower precision. We therefore introduce a layer-aware threshold adjustment that makes precision selection increasingly conservative for sensitive layers. For each layer, we first derive the base threshold from its calibration score distribution:

$$\bar{\tau}^{(l)} = \mathrm{Median}\Big(\mathcal{S}^{(l)}\Big), \tag{13}$$

where $\mathcal{S}^{(l)}$ denotes the calibration scores collected at layer l. To adjust the threshold without an additional layer-wise search, we leverage the layer-wise variation in median importance scores. As illustrated in Fig. 7, the median score generally increases with layer depth, exhibiting a pattern similar to the observed precision sensitivity. We therefore define:

$$\alpha^{(l)} = \alpha^{(l-1)} - R \cdot \frac{\Delta^{(l)}}{\sum_{j=2}^{L} \left|\Delta^{(j)}\right|}, \quad \Delta^{(l)} = \tilde{\mu}^{(l)} - \tilde{\mu}^{(l-1)}, \tag{14}$$

where $\tilde{\mu}^{(l)}$ is the median calibration score and R controls the total adjustment range. The normalized difference determines both the amount and direction of adjustment—an increase in the median score decreases α^l, while a decrease increases it. This produces a smooth threshold variation across adjacent layers and progressively lowers the threshold toward more sensitive deeper layers. The final layer-specific threshold is $\tau^{(l)} = \alpha^{(l)}\,\bar{\tau}^{(l)}$. Consequently, RMP applies lower-precision processing more conservatively in sensitive layers while preserving runtime-adaptive precision selection.

5 Experiments

5.1 Settings

Models and Benchmarks. We evaluate BMoEQ on Mixtral-8×7B-v0.1 (Jiang et al., 2024), Qwen1.5-MoE-A2.7B (Bai et al., 2023), and Qwen3-30B-A3B (Yang et al., 2025). For general-domain evaluation, we report perplexity on WikiText-2 (Merity et al., 2016). We also evaluate zero-shot accuracy on PIQA (Bisk et al., 2019), ARC-Easy/Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), BoolQ (Clark et al., 2019), and OpenBookQA (Mihaylov et al., 2018). We assess multi-domain knowledge and reasoning on MMLU (Hendrycks et al., 2021). For domain-specific evaluation, we measure mathematical reasoning performance on GSM8K (Cobbe et al., 2021) and code-generation performance on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).

Implementation Details. We calibrate using 512 randomly sampled C4 (Raffel et al., 2023) sequences of 2,048 tokens. Routed and shared expert weights are quantized to 2- and 3-bit base precisions with a group size of 128, while all other weights remain in BF16. We select 128 salient channels per 1,024 channels (at least 128 per expert) and assign one additional bit plane, adding 0.125 bits per weight—the same overhead as a 16-bit scaling factor per group. Model-specific configurations are given in Appendix A.4. RMP selects from $\{1, 2\}$ and $\{2, 3\}$ bits for the 2-bit and 3-bit models, respectively. All experiments use an NVIDIA RTX PRO 6000 Max-Q GPU.

Compared Methods. We compare BMoEQ with GPTQ (Frantar et al., 2023), BCQ (Park et al., 2026a), and the MoE-specific method MoEQuant (Hu et al., 2025) as uniform-precision baselines, and PMQ (Huang et al., 2025) and GEMQ (Deng et al., 2026) as static expert-level mixed-precision baselines. We use their official implementations and restrict quantization to expert weights. For static mixed-precision allocation under a 3-bit average budget, we use the candidate bit-width set $\{1, 2, 3, 4\}$, as it yields higher accuracy than $\{2, 3, 4\}$ in our experiments.

5.2 Comparison with Existing MoE Quantization Methods

General- and Multi-Domain Evaluation. Tab. 1 compares BMoEQ with uniform- and mixed-precision expert quantization methods under 2-bit (E2) and 3-bit (E3) settings. While E3 largely preserves performance across methods, E2 causes substantial degradation in uniform methods. BMoEQ mitigates this loss by compensating salient-weight errors while retaining uniform expert-level quantization, with only marginal overhead over BCQ—at most 6.2% in storage and 8.3% in kernel latency. It also outperforms static expert-level mixed-precision methods on most tasks, with particularly pronounced gains at E2; for example, on Qwen3 at E2, BMoEQ achieves 72.55% MMLU accuracy, compared with 63.71% for PMQ and 61.69% for GEMQ. Notably, GEMQ achieves unusually strong perplexity on Qwen3 due to router fine-tuning on the calibration distribution, but this

Table 1: Evaluation on general-domain tasks. Effective weight bits (stored/active) and speedup are measured on Mixtral-8×7B-v0.1. * denotes BMoEQ with RMP applied.

#Bits	Method	Base Quant.	#Eff. w-bits stored (↓)	#Eff. w-bits active (↓)	Speedup (↑)	Mixtral-8×7B-v0.1 PPL (↓)	Mixtral-8×7B-v0.1 0-shot (↑)	Mixtral-8×7B-v0.1 MMLU (↑)	Qwen1.5-MoE-A2.7B PPL (↓)	Qwen1.5-MoE-A2.7B 0-shot (↑)	Qwen1.5-MoE-A2.7B MMLU (↑)	Qwen3-30B-A3B PPL (↓)	Qwen3-30B-A3B 0-shot (↑)	Qwen3-30B-A3B MMLU (↑)
BF16	baseline	-	16	16	×1	3.84	74.56	68.01	7.22	66.08	60.82	8.70	71.01	77.80
E3	GPTQ	Uni.	**3.25**	3.25	×0.6	4.29	72.83	66.33	7.84	64.05	57.91	9.13	69.30	75.66
	BCQ	Uni.	3.38	3.38	×4.15	4.33	72.86	66.24	7.94	64.87	57.61	9.46	70.26	76.47
	MoEQ	Uni.	**3.25**	3.25	×0.6	4.52	73.44	**66.59**	8.06	64.39	58.13	N/A†	N/A†	N/A†
	PMQ	Mixed	**3.25**	3.25	×0.6	4.57	71.90	64.33	7.96	63.87	55.58	9.26	69.91	73.75
	GEMQ				×2.54	**4.20**	72.52	66.04	**7.38**	**65.22**	57.64	**7.42**	69.06	73.49
	BMoEQ	Uni.	3.52	3.52	×3.85	4.26	**74.04**	66.51	7.70	64.98	**58.32**	9.31	**70.44**	**76.63**
	BMoEQ*		3.52	**3.03**	**×4.32**	4.31	73.63	65.98	7.71	65.14	58.28	9.40	70.23	76.49
E2	GPTQ	Uni.	**2.25**	2.25	×0.63	6.78	59.14	33.64	15.79	47.66	30.11	12.42	53.88	26.46
	BCQ				×5.62	6.12	41.98	39.19	14.17	55.37	45.47	12.47	67.93	71.51
	MoEQ	Mixed	**2.25**	2.25	×0.63	9.84	68.74	58.40	15.74	57.71	47.27	N/A†	N/A†	N/A†
	PMQ				×0.63	5.73	65.50	45.13	10.36	57.89	40.13	13.78	65.31	63.71
	GEMQ				×2.92	5.76	63.61	51.82	**8.67**	62.08	44.89	**8.05**	65.55	61.69
	BMoEQ	Uni.	2.39	2.39	×5.19	**5.24**	**71.15**	**59.22**	9.55	**62.20**	**53.05**	11.46	**67.96**	**72.55**
	BMoEQ*		2.39	**1.88**	**×6.4**	5.53	70.19	55.13	9.68	62.05	51.72	11.88	67.19	72.24

† MoEQuant does not release EBSS calibration data or its sampling code for Qwen3.

Table 2: Evaluation on domain-specific generation tasks. We report GSM8K (strict-match), HumanEval (pass@1), and MBPP (pass@1) for three MoE models under E3 and E2 quantization.

#Bits	Method	Mixtral-8×7B-v0.1 GSM8K strict(↑)	Mixtral-8×7B-v0.1 HumanEval pass@1(↑)	Mixtral-8×7B-v0.1 MBPP pass@1(↑)	Qwen1.5-MoE-A2.7B GSM8K strict(↑)	Qwen1.5-MoE-A2.7B HumanEval pass@1(↑)	Qwen1.5-MoE-A2.7B MBPP pass@1(↑)	Qwen3-30B-A3B GSM8K strict(↑)	Qwen3-30B-A3B HumanEval pass@1(↑)	Qwen3-30B-A3B MBPP pass@1(↑)
BF16	baseline	59.59	35.37	48.40	16.76	33.54	39.20	89.46	31.71	71.20
E3	GPTQ	53.45	**37.20**	40.20	22.14	24.39	28.00	88.40	33.54	51.20
	BCQ	51.33	29.27	38.60	17.89	27.44	30.00	**89.23**	**51.83**	70.00
	MoEQ	54.97	34.76	42.80	27.45	**32.32**	33.60	N/A	N/A	N/A
	PMQ	49.13	21.95	36.40	**45.64**	15.85	23.00	83.02	0.00	5.60
	GEMQ	49.89	34.15	35.00	23.96	26.22	28.60	82.87	0.61	6.60
	BMoEQ	**55.12**	28.66	40.20	20.32	27.44	**35.20**	88.70	47.56	**71.60**
	BMoEQ*	51.71	28.66	**43.00**	20.77	26.22	34.60	88.55	48.78	69.40
E2	GPTQ	11.83	0.00	5.80	0.15	0.00	0.00	19.18	0.00	0.00
	BCQ	7.58	5.49	5.00	1.21	2.44	0.60	82.79	**32.93**	61.60
	MoEQ	28.58	21.34	0.20	**21.15**	16.46	18.40	N/A	N/A	N/A
	PMQ	20.24	1.22	0.00	10.39	1.83	0.00	74.30	0.00	2.00
	GEMQ	8.72	3.05	3.20	18.42	7.32	9.00	57.54	0.00	0.20
	BMoEQ	**37.15**	**23.17**	**23.40**	20.55	**17.07**	20.20	**84.46**	31.10	**66.40**
	BMoEQ*	31.92	18.90	21.80	14.40	13.41	**21.80**	83.47	30.49	65.00

gain does not transfer to zero-shot or MMLU accuracy, indicating calibration-domain overfitting. With RMP, the active effective bit-width decreases from 3.52 to 3.03 at E3 and from 2.39 to 1.88 at E2, yielding $4.32\times$ and $6.40\times$ kernel speedups over BF16 while still maintaining higher zero-shot and MMLU accuracy than the static mixed-precision baselines in most settings.

Evaluation on Domain-Specific Generation Tasks. Table 2 evaluates BMoEQ on mathematical reasoning and code-generation tasks. Although most methods remain competitive at E3, the accuracy of the compared methods degrades sharply at E2. In contrast, BMoEQ consistently preserves stronger domain-specific performance across most models and tasks; for example, on E2 Mixtral, it achieves 37.15 on GSM8K, 23.17 on HumanEval, and 23.4 on MBPP, substantially outperforming PMQ and GEMQ. This advantage highlights the benefit of avoiding calibration-fixed expert bit allocation for domains underrepresented during calibration. With RMP, BMoEQ further lowers runtime precision while retaining competitive domain-specific accuracy.

Accuracy Comparison at Matched Effective Bit-Widths. In Tab. 1, BMoEQ uses a slightly larger storage budget than the static mixed-precision baselines, with approximately 8.3% and 6.2% overhead at E3 and E2, respectively. To isolate the benefit of our quantization scheme from this budget difference, we adjust the bit allocation of PMQ and GEMQ to match the effective bit-widths of BMoEQ (3.52 and 2.39) and report the results in Tab. 3. Under this matched budget, BMoEQ outperforms both baselines on all reported accuracy at E2, except perplexity, where GEMQ benefits from calibration-domain router fine-tuning. At E3, expert-only quantization largely preserves model performance across all methods. These results show that BMoEQ gains from using the bit budget more effectively, rather than simply using more bits: prioritizing salient weights within uniformly quantized experts outperforms static expert-level bit allocation.

Table 3: Comparison with static expert-level mixed-precision methods at matched effective bits. PMQ and GEMQ are matched to our effective stored bits per weight at each setting.

#Bits	Method	Eff. stored	Mixtral-8×7B-v0.1 PPL (↓)	0-shot (↑)	MMLU (↑)	GSM8K (↑)	Qwen1.5-MoE-A2.7B PPL (↓)	0-shot (↑)	MMLU (↑)	GSM8K (↑)	Qwen3-30B-A3B PPL (↓)	0-shot (↑)	MMLU (↑)	GSM8K (↑)
BF16	baseline	16	3.84	74.56	68.01	59.59	7.22	66.08	60.82	16.76	8.70	71.01	77.80	89.46
E3	PMQ	3.52	4.36	73.11	64.14	49.66	7.85	63.98	57.36	**44.28**	9.06	69.88	74.26	83.78
	GEMQ		**4.16**	72.81	**66.63**	51.78	**7.37**	**65.28**	58.06	25.78	**7.41**	70.38	74.22	83.47
	BMoEQ	3.52	4.26	**74.04**	66.51	**55.12**	7.70	64.98	**58.32**	20.32	9.31	**70.44**	**76.63**	**88.70**
E2	PMQ	2.39	5.48	63.78	47.63	24.41	9.64	58.95	40.82	18.73	12.11	66.21	71.11	76.27
	GEMQ		5.42	67.98	55.78	15.47	**8.56**	61.22	48.00	19.94	**7.84**	67.70	66.77	65.81
	BMoEQ	2.39	**5.24**	**71.15**	**59.22**	**37.15**	9.55	**62.20**	**53.05**	**20.55**	11.46	**67.96**	**72.55**	**84.46**

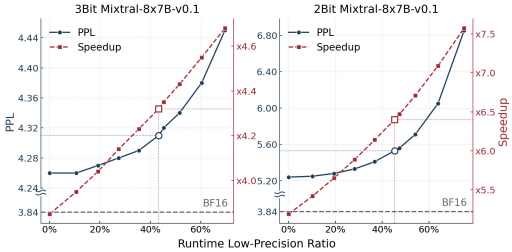

Figure 4: PPL and speedup trade-off across runtime low-precision ratios on Mixtral-8×7B-v0.1 (left: 3-bit, right: 2-bit).

#Bits	Method	Peak Mem.(GB)	Saving	Token/sec	Speed-up
FP16	Baseline	94.16	×1	51.04	×1
E3	BCQ	23.52	×4.00	114.75	×2.25
	BMoEQ	25.30	×3.72	112.34	×2.20
	BMoEQ*			117.17	×2.30
E2	BCQ	17.62	×5.34	128.25	×2.51
	BMoEQ	19.40	×4.85	125.88	×2.47
	BMoEQ*			131.60	×2.58

Table 4: End-to-end GPU performance on Mixtral-8×7B-v0.1 (batch size 1 and 128 generated tokens).

5.3 GPU Performance

RMP Speedup. Fig. 4 shows the accuracy–latency trade-off of RMP by varying the threshold to control the runtime low-precision ratio from 0% to 80%. As more bit-planes are skipped, the expert linear-layer speedup increases nearly linearly, whereas perplexity degrades non-linearly and begins to increase more rapidly beyond a certain ratio. We therefore use the median calibration score as the default threshold, which lies near the onset of this sharper degradation. At this operating point, RMP limits the PPL increase to only 0.05/0.29 for the 3-bit/2-bit models while achieving $\times 4.32$, $\times 6.40$ kernel speedup, respectively. Since the low-precision ratio is controlled solely by the runtime threshold, the same quantized model can flexibly trade accuracy for latency without requantization.

End-to-end Evaluation. Despite targeting only the expert layers, BMoEQ reduces peak memory from 94.16 GB to 25.3 GB and 19.4 GB at E3 and E2, corresponding to $3.72\times$ and $4.85\times$ memory savings(Tab. 4). Additionally, optimizing the expert linear operations yields $2.2\times$ and $2.47\times$ end-to-end decoding speedups, respectively. Although the fused saliency-aware BCQ path incurs a slight latency overhead over vanilla BCQ, RMP further improves the speedups to $2.30\times$ and $2.58\times$, surpassing BCQ by reducing runtime expert computation. These gains require no changes to attention, routing, KV-cache, or other framework components. The larger $4.3\times$ / $6.4\times$ kernel-level speedups are therefore bounded at the model level by the remaining.

6 Conclusion

We presented BMoEQ, a quantization method for MoE-LLMs that combines uniform ultra-low-bit base precision across experts with input-adaptive execution precision through the bit-plane structure of BCQ. Saliency-aware error reconstruction assigns a single additional bit-plane to a small subset of salient weights, recovering accuracy without static expert-level bit allocation, while Runtime-Adaptive Mixed-Precision (RMP) selectively omits bit-planes for less important activated experts, enabling a single stored representation to support multiple execution precisions without requantization. In our evaluation, BMoEQ demonstrates superior overall performance, with its advantages most pronounced in the 2-bit ultra-low-bit regime and on domain-specific tasks. With our fused GPU kernels, RMP achieves up to $6.4\times$ kernel-level and $2.58\times$ end-to-end speedups over the BF16 baseline, demonstrating that jointly optimizing low-bit expert representations, runtime precision, and GPU execution yields efficient MoE-LLM serving.

AI USE STATEMENT

In this paper, we used a large language model to aid and polish the manuscript to improve the overall writing quality. All AI-assisted edits were reviewed by the authors, and we take full responsibility for the final paper.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

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, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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, 2026. URL https://arxiv.org/abs/2605.23078.

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, 2025. URL https://arxiv.org/abs/2505.05799.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

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, 2025. URL https://arxiv.org/abs/2505.03804.

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, 2025. URL https://arxiv.org/abs/2410.06270.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction, 2021. URL https://arxiv.org/abs/2102.05426.

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, 2024. URL https://arxiv.org/abs/2402.14800.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2409.02060.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models, 2024. URL https://arxiv.org/abs/2206.09557.

Gunho Park, Jeongin Bae, Beomseok Kwon, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. Anybcq: Hardware efficient flexible binary-coded quantization for multi-precision llms, 2026a. URL https://arxiv.org/abs/2510.10467.

Yeonsik Park, Hyeonseong Kim, and Seungkyu Choi. Serq: Saliency-aware low-rank error reconstruction for llm quantization, 2026b. URL https://arxiv.org/abs/2603.08185.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

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, 2017. URL https://arxiv.org/abs/1701.06538.

Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks, 2018. URL https://arxiv.org/abs/1802.00150.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. Lqer: Low-rank quantization error reconstruction for llms, 2024. URL https://arxiv.org/abs/2402.02446.

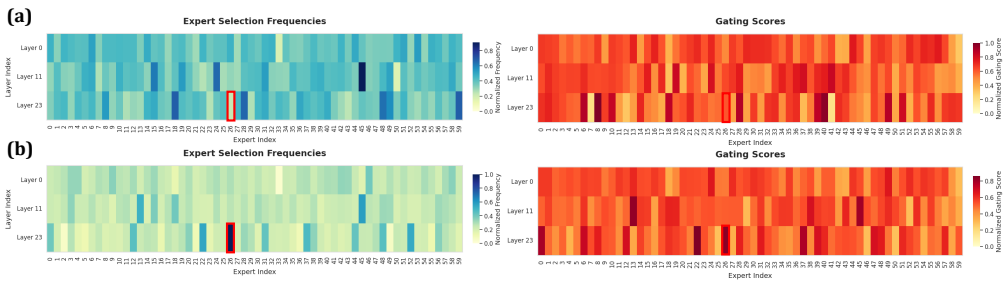

Figure 5: Distribution of the key terms constituting expert importance. We compare expert selection frequencies and gating scores across different domain tasks on Qwen1.5-MoE-A2.7B: (a) mathematical reasoning (GSM8K) and (b) general-domain corpus (WikiText-2).

A Appendix

A.1 Expert Importance Across Downstream Domains

Existing static expert-level mixed-precision methods() determine each expert's bit-width offline using calibration-derived statistics, such as expert selection frequency and gating scores, as proxies for expert importance. Since these statistics are typically collected from a small general-domain calibration dataset, the resulting expert-importance estimates can be biased toward the calibration domain. We therefore compare the routing statistics of Qwen1.5-MoE-A2.7B between the mathematical reasoning benchmark GSM8K and the general-domain WikiText-2 dataset to characterize this domain dependence. As shown in Fig. 5, the resulting expert-importance distributions differ substantially across the two domains. Experts that are frequently selected or assigned high routing scores on WikiText-2 can remain largely inactive on GSM8K. For example, expert 26 in layer 23 exhibits markedly different behavior between the two datasets. This observation indicates that expert importance is not a fixed property of the model but varies substantially across input domains. Consequently, static bit allocation based on general-domain calibration can assign insufficient precision to experts that become important on domain-specific inputs, leading to larger downstream accuracy degradation. This limitation motivates runtime-adaptive precision selection based on expert importance observed during inference.

A.2 Truncation Error Analysis Between Integer and BCQ Format

We analyze the additional error introduced by runtime precision truncation, rather than the original quantization error. Since truncation acts independently on each weight, we consider a single weight element w and reduce its B-bit representation to $(B-1)$ bits by discarding the least significant component. For conventional integer quantization, the quantized weight is represented as

$$\widehat{w}_{\text{INT}} = \eta \cdot s \cdot \sum_{j=0}^{B-2} 2^j \cdot c_j, \qquad c_j \in \{0,1\}, \quad \eta \in \{-1,+1\}, \tag{15}$$

where s denotes the quantization scaling factor and η denotes the sign. Dropping the least significant bit (LSB) gives

$$\widehat{w}_{(B\to B-1)} = \eta \cdot s \cdot \sum_{j=1}^{B-2} 2^j \cdot c_j, \tag{16}$$

and hence the truncation error is

$$e^{\text{INT}}_{\text{trunc}} = \widehat{w}_{\text{INT}} - \widehat{w}_{(B\to B-1)} = \eta \cdot s \cdot c_0. \tag{17}$$

Assuming approximately balanced LSB occupancy, i.e., $P(c_0 = 1) \approx 1/2$, the expected squared truncation error becomes

$$\mathbb{E}\Big[\big(e_{\text{trunc}}^{\text{INT}}\big)^2\Big] \approx \frac{s^2}{2}. \tag{18}$$

In contrast, BCQ progressively approximates the weight using successive bit-planes,

$$\widehat{w}_{\text{BCQ}} = \sum_{i=1}^{B} \alpha_i \cdot b_i, \qquad b_i \in \{-1, +1\}. \tag{19}$$

When the final refinement plane is removed, the truncation error and expected squared truncation error are simply

$$e_{\text{trunc}}^{\text{BCQ}} = \alpha_B \cdot b_B, \qquad \mathbb{E}\left[\left(e_{\text{trunc}}^{\text{BCQ}}\right)^2\right] = \alpha_B^2. \tag{20}$$

The key difference lies in how s and α_B are determined. Extending the scalar analysis above to a quantization group $\boldsymbol{w} \in \mathbb{R}^g$, the scale in max-scaled integer quantization is determined by the largest weight in the group:

$$s = \frac{\max_i |w_i|}{2^{B-1} - 1}, \tag{21}$$

whereas in BCQ the final coefficient is fitted to the residual remaining after the preceding bit-planes,

$$\boldsymbol{r} = \boldsymbol{w} - \sum_{j=1}^{B-1} \alpha_j \cdot \boldsymbol{b}_j, \qquad \alpha_B = \arg\min_{\alpha} \|\boldsymbol{r} - \alpha \cdot \boldsymbol{b}_B\|_2^2, \tag{22}$$

Thus, s is tied to the original weight range, whereas α_B is determined from an already refined residual, which is typically much smaller than the range of $\boldsymbol{w} \in \mathbb{R}^g$, where g is the quantization group size. For an intuitive comparison, assume $w_i \sim \mathcal{N}(0, \sigma^2)$ and the group size of $N = 128$. Using the leading-order approximation (i.e., $\max_i |w_i| \approx \sigma\sqrt{2 \ln N}$), 3-bit integer quantization gives approximately

$$\mathbb{E}\Big[\big(e_{\text{trunc}}^{\text{INT}}\big)^2\Big] \approx 0.5\,\sigma^2. \tag{23}$$

Under progressive BCQ fitting, the final 3-bit coefficient is approximately $\alpha_3 \approx 0.27\sigma$, yielding

$$\mathbb{E}\left[\left(e_{\text{trunc}}^{\text{BCQ}}\right)^2\right] \approx 0.073\,\sigma^2. \tag{24}$$

Thus, under this simplified Gaussian model, BCQ introduces a substantially smaller perturbation when reducing precision from 3 to 2 bits. This illustrates why the progressively refined bit-plane representation of BCQ is particularly suitable for runtime precision selection.

A.3 Layer-wise Sensitivity to Runtime Low-Precision Ratio

To characterize layer-wise sensitivity to runtime precision reduction, we sweep the low-precision ratio from 10% to 80% for one contiguous layer group at a time, while keeping all remaining layers at 3-bit precision. We divide Mixtral-8×7B-v0.1 into four groups of eight layers and Qwen3-30B-A3B into four groups of twelve layers, and evaluate perplexity on WikiText-2 (Fig. 6). In both models, the final layer group exhibits the highest sensitivity: Mixtral layers 24–31 increase from 4.29 to 4.38 perplexity at a $\sim 72\%$ low-precision ratio, while Qwen3 layers 36–47 increase from 9.32 to 9.74 at $\sim 80\%$; earlier groups show substantially smaller degradation. The degradation is also non-linear, remaining limited up to approximately 40–50% before increasing sharply at higher low-precision ratios, especially in the final layer group. These results motivate a layer-aware threshold adjustment that progressively lowers the threshold toward deeper layers, making low-precision selection more conservative where sensitivity is higher.

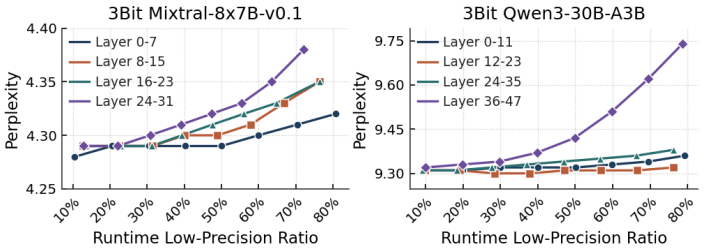

Figure 6: Layer-wise sensitivity to runtime low-precision ratio for 3-bit Mixtral-8×7B-v0.1 (left) and Qwen3-30B-A3B (right).

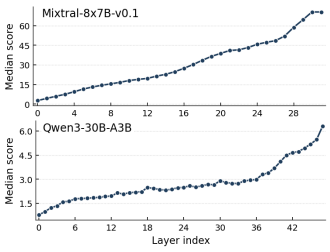

Figure 7: Layer-wise median RMP score, increasing with depth.

A.4 Salient-Channel Configuration across Model

Following the rule in experiment settings (128 salient intermediate channels per 1,024, with at least 128 per expert), Table 5 lists the resulting number of salient channels for the routed experts (E) and shared experts (SE) of each model.

Table 5: Salient-channel configuration. d_h is the hidden size, d_i the MoE intermediate size; one bit-plane is added to 128 salient channels per 1,024 intermediate channels.

		d_i		#Salient channels	
Model	d_h	E	SE	E	SE
Mixtral-8×7B-v0.1	4,096	14,336	N/A	1,792	N/A
Qwen1.5-MoE-A2.7B	2,048	1,408	5,632	128	768
Qwen3-30B-A3B	2,048	768	N/A	128	N/A

A.5 Detailed Accuracy Results

Tab. 6 reports the detailed results underlying Tab. 1, including WikiText-2 perplexity, seven zero-shot reasoning tasks, and MMLU across four subject categories. MoEQuant results for Qwen3-30B-A3B are reported as N/A because its EBSS calibration data and sampling code are not publicly available.

Table 6: Detailed results on 0-shot commonsense reasoning and MMLU for three MoE models under E3 and E2 quantization.

Model	#Bits	Methods	PPL (↓)	0-Shot Common Sense Reasoning tasks								MMLU				
				BoolQ (↑)	PIQA (↑)	ARC-e (↑)	ARC-c (↑)	HellaS. (↑)	WinoG. (↑)	OBQA (↑)	Avg. (↑)	Human. (↑)	Other (↑)	SocialS. (↑)	STEM (↑)	Avg. (↑)
Mixtral -8x7B-v0.1	BF16	Baseline	3.84	86.06	83.13	82.49	59.73	84.18	76.95	49.40	74.56	60.55	75.73	79.98	59.85	68.01
	E3	GPTQ	4.29	82.29	82.59	80.18	57.08	**83.12**	75.14	**49.40**	72.83	59.49	73.29	77.80	58.48	66.33
		BCQ	4.33	83.33	82.64	81.86	57.42	82.71	75.69	46.40	72.86	60.04	**73.64**	77.09	57.63	66.24
		MoEQ	4.52	84.37	82.54	81.44	57.25	82.48	**77.58**	48.40	73.44	**60.26**	73.32	77.09	**59.15**	**66.59**
		PMQ	4.57	83.79	82.10	78.41	55.46	82.19	73.95	47.40	71.90	57.77	70.97	75.85	56.33	64.33
		GEMQ	**4.20**	81.56	82.10	81.02	57.51	82.83	76.24	46.40	72.52	59.36	73.45	77.19	57.82	66.04
		BMoEQ	4.26	**84.98**	82.75	**82.87**	**59.13**	82.85	76.48	49.20	**74.04**	60.00	**73.64**	**77.93**	58.07	66.51
		BMoEQ*	4.31	84.71	**82.86**	81.57	58.96	82.34	76.16	48.80	73.63	59.40	73.32	77.58	57.25	65.98
	E2	GPTQ	6.78	67.34	74.48	58.46	38.23	70.47	65.59	39.40	59.14	33.69	35.31	35.94	29.69	33.64
		BCQ	6.12	61.56	68.99	32.07	27.13	25.89	51.85	26.40	41.98	34.22	47.57	41.53	36.06	39.19
		MoEQ	9.84	**84.86**	78.45	75.00	50.85	74.56	**75.06**	42.40	68.74	52.99	65.59	68.18	**49.86**	58.40
		PMQ	5.73	79.72	78.35	72.31	46.84	71.73	69.77	39.80	65.50	39.98	52.78	51.45	39.11	45.13
		GEMQ	5.76	75.60	77.42	70.79	43.86	69.05	69.53	39.00	63.61	47.31	59.16	59.28	44.05	51.82
		BMoEQ	**5.24**	80.86	**81.66**	**81.40**	**54.18**	**79.27**	73.88	**46.80**	**71.15**	**53.54**	**67.27**	**70.23**	49.03	**59.22**
		BMoEQ*	5.53	81.01	81.07	79.38	53.50	77.17	74.03	45.20	70.19	50.07	64.37	64.87	44.08	55.13

Model	#Bits	Methods	PPL (↓)	0-Shot Common Sense Reasoning tasks								MMLU				
				BoolQ (↑)	PIQA (↑)	ARC-e (↑)	ARC-c (↑)	HellaS. (↑)	WinoG. (↑)	OBQA (↑)	Avg. (↑)	Human. (↑)	Other (↑)	SocialS. (↑)	STEM (↑)	Avg. (↑)
Qwen1.5 -MoE-A2.7B	BF16	Baseline	7.22	79.82	80.20	68.98	43.69	77.30	68.75	43.80	66.08	55.13	68.43	70.36	52.52	60.82
	E3	GPTQ	7.84	75.29	79.76	65.87	43.26	75.42	66.77	42.00	64.05	52.52	64.79	67.40	49.92	57.91
		BCQ	7.94	77.61	79.82	68.10	42.75	75.24	69.38	41.20	64.87	52.56	66.46	66.30	47.95	57.61
		MoEQ	8.06	76.94	79.76	66.50	42.24	74.59	68.11	42.60	64.39	52.01	66.14	**67.47**	**50.27**	58.13
		PMQ	7.96	72.26	79.16	66.96	43.26	75.84	68.43	41.20	63.87	51.12	64.44	63.93	45.35	55.58
		GEMQ	**7.38**	75.81	**80.58**	67.59	**43.60**	**76.20**	68.98	**43.80**	**65.22**	52.31	65.72	67.08	48.43	57.64
		BMoEQ	7.70	**78.62**	79.54	67.97	42.06	75.77	69.53	41.40	64.98	52.77	**67.20**	67.05	49.32	**58.32**
		BMoEQ*	7.71	78.53	79.71	**68.22**	42.66	75.48	**69.61**	41.80	65.14	**53.11**	67.07	66.98	48.81	58.28
	E2	GPTQ	15.79	59.60	66.05	42.93	28.24	53.57	54.06	29.20	47.66	29.82	33.41	30.09	27.31	30.11
		BCQ	14.17	62.63	72.52	52.69	33.19	66.52	62.67	37.40	55.37	41.40	53.01	51.87	37.87	45.47
		MoEQ	15.74	71.22	75.41	63.01	35.92	62.38	61.80	34.20	57.71	42.51	55.07	54.11	39.99	47.27
		PMQ	10.36	69.39	75.35	55.60	34.64	66.99	64.48	38.80	57.89	34.52	46.41	47.48	35.14	40.13
		GEMQ	**8.67**	**76.12**	76.99	**66.33**	**40.53**	69.19	**66.38**	39.00	62.08	42.89	51.75	50.50	35.65	44.89
		BMoEQ	9.55	72.63	78.24	66.16	39.59	**70.63**	65.75	**42.40**	**62.20**	**48.10**	**60.89**	**61.46**	**44.50**	**53.05**
		BMoEQ*	9.68	73.27	**78.67**	65.15	39.33	70.15	65.98	41.80	62.05	47.25	59.03	59.60	43.48	51.72

Model	#Bits	Methods	PPL (↓)	0-Shot Common Sense Reasoning tasks								MMLU				
				BoolQ (↑)	PIQA (↑)	ARC-e (↑)	ARC-c (↑)	HellaS. (↑)	WinoG. (↑)	OBQA (↑)	Avg. (↑)	Human. (↑)	Other (↑)	SocialS. (↑)	STEM (↑)	Avg. (↑)
Qwen3 -30B-A3B	BF16	Baseline	8.70	88.84	80.03	79.63	56.23	77.61	70.32	44.40	71.01	67.86	81.62	87.72	79.19	77.80
	E3	GPTQ	9.13	86.70	**80.25**	76.43	51.88	75.98	70.88	43.00	69.30	65.14	80.17	86.16	76.66	75.66
		BCQ	9.46	87.49	79.38	78.96	**55.12**	75.84	71.03	44.00	70.26	**67.10**	80.59	85.80	77.29	76.47
		MoEQ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		PMQ	9.26	87.92	79.60	76.77	54.44	77.01	69.06	44.60	69.91	65.74	79.79	85.21	68.57	73.75
		GEMQ	**7.42**	**88.29**	79.60	71.59	50.68	**77.91**	**71.35**	44.00	69.06	66.21	80.08	83.98	67.65	73.49
		BMoEQ	9.31	87.49	79.27	**79.12**	55.03	75.95	70.64	45.60	**70.44**	66.65	**81.07**	86.16	77.86	**76.63**
		BMoEQ*	9.40	87.46	79.16	79.00	53.67	75.97	70.32	**46.00**	70.23	66.23	80.79	**86.29**	**78.02**	76.49
	E2	GPTQ	12.42	71.99	71.38	48.95	32.85	63.33	57.06	31.60	53.88	26.89	27.97	26.00	24.80	26.46
		BCQ	12.47	86.67	78.13	**76.85**	**51.88**	69.99	**70.40**	41.60	67.93	60.87	75.89	81.80	73.04	71.51
		MoEQ	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		PMQ	13.78	79.42	**79.11**	69.82	47.10	73.57	67.72	40.40	65.31	57.53	71.39	77.38	52.01	63.71
		GEMQ	**8.05**	86.30	77.91	66.25	42.92	**74.94**	68.35	42.20	65.55	57.19	68.52	73.94	49.73	61.69
		BMoEQ	11.46	**86.91**	78.51	76.18	51.11	70.91	70.09	42.00	**67.96**	**61.08**	**78.05**	**82.42**	**74.63**	**72.55**
		BMoEQ*	11.88	86.57	78.13	74.20	48.63	70.99	69.22	**42.60**	67.19	60.81	77.37	82.32	74.41	72.24