THE BLIND SPOT OF POST-TRAINING QUANTIZATION IN MIXTURE-OF-EXPERTS MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In mixture-of-experts (MoE) models, quantization changes which experts each token is routed to. Post-training quantization (PTQ) methods minimize the reconstruction error of each layer, and prior work evaluates the quantized models with benchmark accuracy or the expert-selection change rate. However, we argue that none of these captures how much deviation the routing changes cause. In this paper, we propose a metric for this routing-induced deviation, which compares the quantized model's deviation from the original model with and without fixing its expert selection to the original one. We further derive a law for this deviation, which shows that the layer-wise reconstruction error misses a growing fraction of the deviation as the bit width increases. Lastly, we apply our metric to a released quantized checkpoint and show that it captures a change that accuracy cannot.

1 INTRODUCTION

Post-training quantization (PTQ) has emerged as a standard way to deploy large language models. PTQ methods are designed so that the outputs of the quantized model stay close to those of the original model, and their success is usually checked with benchmark accuracy. In practice, they minimize the reconstruction error, which measures how far the output of each linear layer departs from that of the original model (Section 2.2); GPTQ and AWQ, two widely used examples, reduce it using second-order information and a search over per-channel scales, respectively (Frantar et al., 2023; Lin et al., 2024). Benchmark accuracy checks whether a quantized checkpoint answers standard benchmarks such as MMLU and GSM8K correctly as often as the original model. For dense models, these two quantities have served as the standard check. However, mixture-of-experts (MoE) models add a router component that dense models lack (Shazeer et al., 2017; Fedus et al., 2022). At every MoE layer, the router selects the top-k experts for each token. This top-k routing is discrete and non-differentiable, so the small error that quantization leaves in the weights can change the expert selection itself. When it does, the model executes a different expert set for that token and departs from the original model's behavior.

Two publicly released quantized checkpoints show what is at stake (JunHowie/Qwen3-30B-A3B-GPTQ-Int4, Qwen/Qwen3-30B-A3B-GPTQ-Int4). Both are 4-bit GPTQ checkpoints of the same 30B-parameter MoE model, with identical quantization settings apart from the router (bit width, group size, symmetry, act-order, and checkpoint format). The community checkpoint quantizes the router weights of all 48 layers to 4 bits, whereas the official release keeps them in full precision. These tensors hold 12,582,912 parameters, or 0.0412% of all weights. Yet on the same input, the checkpoint with quantized router weights sends 75.9% of tokens to a different expert set than the original model, compared with 43.9% for the other (131,008 WikiText-2 tokens of Section 3.3, median across the 48 layers; Appendix C). Despite this difference, whether and how to quantize the router weights varies across toolkits, versions, and methods (GPTQModel PR #1569, 2025; GPTQModel PR #1995, 2025; GPTQModel v5.0.0, 2025; AutoAWQ, 2025; llama.cpp, 2026; mlx-lm, 2026). We found no consensus on whether to quantize the router across nine quantization toolkits for seven MoE architectures or across 300 quantized variants of ten MoE models on the Hugging Face Hub. Deciding how to quantize the router therefore first requires measuring how much deviation changes in expert selection produce at the output (Section 4.4; Appendices C and D). This raises a question:

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.

Do the reconstruction error and benchmark accuracy capture the size of the deviation that arises when the expert selection changes?

Prior work has reported that quantization changes routing (Chen et al., 2025; Fu et al., 2025). These two quantities, however, do not capture changes in expert selection. The reconstruction error that GPTQ and AWQ minimize compares the output of one weight matrix at a time with that of the original. It therefore does not reflect changes in which experts are selected (Park et al., 2026a; 2026b). For benchmark accuracy, Dutta et al. (2024) showed on dense models that it does not reveal whether a quantized model behaves like the original model, and proposed distance metrics such as the KL divergence. The KL divergence, however, requires a probability distribution, which a language model defines only at its output, so it does not tell at which layer or through which path a deviation arises. Instead, routing-change metrics such as the expert-selection change rate (Chen et al., 2025; Deng et al., 2026) and top-k overlap (Park et al., 2026a; 2026b) count routing changes directly at every layer, but only how often the expert selection changes, not how much deviation those changes produce.

In this work, we propose R, a metric of the routing-induced deviation that two runs of the quantized model separate from the deviation that the reconstruction error captures (Section 3). The measurement requires only inference, with no additional training or architectural changes. We define the blind fraction as the share of the deviation that the reconstruction error cannot capture. Experiments on seven publicly released MoE models confirm that, at sufficiently high bit widths, the blind fraction grows as the bit width increases. Finally, we measure the routing-induced deviation directly and compare it with the expert-selection change rate across models and bit widths, finding that the deviation caused by a single change in expert selection varies across both.

2 Background and Related Work

2.1 Mixture of Experts and the Router

An MoE layer splits the feed-forward layer of a dense model into multiple experts and computes only a few of them for each token (Shazeer et al., 2017; Fedus et al., 2022). Let u denote the input to the layer. The router uses its weights W_g to compute a score vector $z = W_g u$, one score per expert, and selects the k experts with the highest scores; $\mathrm{TopK}(z, k)$ returns the indices of the k largest entries of z, and $T(u)$ is this set. The layer output y is the sum of the outputs of the selected experts, weighted by their gate values:

$$T(u) = \mathrm{TopK}(W_g u,\ k), \qquad y(u) = \sum_{e \in T(u)} g_e(u)\, E_e(u). \tag{1}$$

Here, $E_e(u)$ is the output of expert e, a feed-forward network applied to u, and the gate value $g_e(u)$ is a weight computed from the router scores; the exact calculation varies across models. The router weights W_g form a single matrix per layer and make up only 0.002–0.022% of the non-embedding parameters in four MoE models (Chen et al., 2025). Yet this small matrix determines two things: which experts a token is sent to (the set T, hereafter the expert selection) and how much each of their outputs contributes (the gate values g). The gate values vary continuously with the scores, whereas the expert selection depends only on the order of the scores.

As a result, even a small shift in the scores can reorder the experts and change the expert selection, and this change also affects the layer output (Park et al., 2026b). This happens even when the router weights are identical to those of the original model, because quantization error in the preceding layers alters the router input u (Fang & Huang, 2026; Lee et al., 2026). Of the two things the router determines, this paper measures the deviation that arises when the expert selection changes.

2.2 Post-Training Quantization

How weights are quantized. Post-training quantization replaces each weight with a value on a finite grid: round-to-nearest (RTN) takes the nearest grid point, whereas GPTQ and AWQ adjust the rounding on a small calibration set so that each layer's output changes as little as possible (Frantar et al., 2023; Lin et al., 2024).

What is minimized. Post-training quantization methods such as GPTQ minimize the output error between a weight matrix W and its quantized counterpart $\hat{W}$ on the inputs X that the matrix receives from the calibration data:

$$\min_{\hat{W}} \|WX - \hat{W}X\|_F^2. \tag{2}$$

We call the value of Eq. (2) the reconstruction error, where $\|\cdot\|_F$ is the Frobenius norm and the minimum is taken over $\hat{W}$ with entries on the quantization grid. It compares only the outputs of the matrix while holding fixed which tokens are routed to which expert, so it does not capture the effect of quantization on the expert selection. Block-wise methods (Shao et al., 2024; Cheng et al., 2024) instead choose the quantized weights to reduce the output error of an entire decoder block, the attention and MoE layers together. Their error does include the effect of changes in the expert selection within the block, but because top-k selection is non-differentiable, the error due to routing differences cannot be reduced through gradients. Moreover, the output of one block is the input to the next, and the resulting change in the next block's routing is not included in this error at all (Park et al., 2026b). For these reasons, Park et al. (2026a; 2026b) point out that minimizing the reconstruction error alone keeps the block outputs close to those of the original model but places no constraint on the expert selection or on the effect of changing it. To address this, we separate the deviation that the reconstruction error captures from the routing-induced deviation (Section 3).

Which weights are quantized. Previous methods differ on whether to quantize the router. EAC-MoE keeps it in full precision because the router is a tiny fraction of the parameters yet important for the expert selection (Chen et al., 2025), whereas EAQuant quantizes both the weights and the activations of the router to 8 bits (Fu et al., 2025). Settling this choice, on which toolkits and papers diverge, requires measuring how much quantizing the router changes the deviation (Section 4; Appendix A).

2.3 Evaluating Quantized Models

Capability and distance metrics. Evaluations of quantized models fall into two categories: capability and distance metrics (Dutta et al., 2024). Capability metrics, such as benchmark accuracy and perplexity, measure how well a model performs a task. Distance metrics measure how far the outputs of the quantized model are from those of the original model. Capability metrics compare against the correct answer, not against the original model: answers that change from correct to incorrect can cancel out answers that change the other way, leaving accuracy unchanged. Dutta et al. measured this on dense models. Instead of using capability metrics alone, they proposed two distance metrics that take the original model as the reference: KL divergence between output distributions and the fraction of answers that change. Still, both were developed on dense models and do not address routing. For MoE models, Sakthivel et al. (2026) likewise report that, with the number of questions they could afford to evaluate, accuracy cannot distinguish the quantization settings they compare. They therefore use the negative log-likelihood (NLL) of the correct answer instead. Capability metrics, including the NLL of the correct answer, are well suited to comparing methods. To ask how much of the deviation comes from routing, however, distance metrics are needed. Taking the original model as the reference, they capture any change in the output distribution, not only changes in the probability of the correct answer.

Measuring routing change. To capture the effect of routing, quantization studies of MoE models have added a third kind of quantity: for the same token, it compares the top-k expert sets selected by the original and quantized models. Methods of measuring routing change differ across studies. In this paper, we use the fraction of tokens for which at least one selected expert changes, and call it the expert-selection change rate (hereafter, the change rate; Chen et al., 2025). We compute it for each layer and take the median across layers to limit the influence of extreme layers. Like the other quantities, the change rate counts only how often the expert selection changes, so it does not tell how much deviation from the original model those changes produce. We measure that deviation by running the quantized model while fixing only its expert selection to that of the original model (Section 3).

3 MEASURING THE ROUTING-INDUCED DEVIATION WITH FREE AND FROZEN RUNS

The reconstruction error of Eq. (2) is computed with the expert selection held fixed (Section 2.2). We therefore fix the expert selection of the quantized model to that of the original model and define the part of the deviation that disappears as the routing-induced deviation, which the reconstruction error cannot capture.

3.1 FREE AND FROZEN RUNS

Two runs. We run the quantized model twice on the same input. In the *free run*, the router of the quantized model selects the experts as usual. Because of quantization error, its selection can differ from that of the original model. In the *frozen run*, we force the quantized model to use the same top-k indices that the original model selects at each token and layer. The gate values are still computed from the scores of the quantized model's router, using the model's own gating function. Let D_{free} and D_{frozen} denote the deviations of the two runs from the original model. We define

$$R = D_{\text{free}} - D_{\text{frozen}}, \qquad m = R/D_{\text{free}}. \tag{3}$$

The deviation D can be any distance metric (Section 2.3); we mainly use two forms, depending on where it is measured (Section 3.2):

$$D = \frac{\sum_t \|\hat{y}_t - y_t\|^2}{\sum_t \|y_t\|^2} \ \text{(layer-local)}, \qquad D = \frac{1}{N}\sum_t \mathrm{KL}\big(p_t \,\|\, \hat{p}_t\big) \ \text{(model-output)}. \tag{4}$$

Here, y_t is the output of an MoE layer at position t (including shared experts, before the residual connection), p_t is the next-token distribution at position t, N is the number of positions, and a hat marks values from the quantized model. In Eq. (3), R is the routing-induced deviation and m is the blind fraction; for example, $m = 0.9$ means that fixing only the expert selection removes 90% of the deviation, and the remaining 10% is the part that the reconstruction error captures.

Why only the indices are fixed. The weight error changes the scores, and the gate values drift continuously with this score change. The reconstruction error of the router weights (O_ℓ in Eq. (5)) measures the score change. The expert selection, in contrast, changes only when an expert enters or leaves the top-k. Since this change is discontinuous, the reconstruction error does not capture it. We therefore fix only the expert selection and leave the gate-value drift in D_{frozen} (Appendix E checks that the forcing adds no other deviation, compares it with prior restoring experiments, and reports the effect of also fixing the gate values).

D_{frozen} tracks the reconstruction error. When only the router weights of layer ℓ are quantized, we write

$$D_{\text{frozen},\ell} = \kappa_\ell \, O_\ell, \qquad O_\ell = \frac{\sum_t \|(\hat{W}_g - W_g)\, u_t\|^2}{\sum_t \|W_g u_t\|^2}, \tag{5}$$

where O_ℓ is the reconstruction error (Eq. (2)) evaluated on the router inputs u_t from our measurement data and normalized by the original router output, and κ_ℓ is a proportionality constant. The constant differs across layers but varies little with the bit width, so at high bit widths we treat it as nearly constant within each layer (Appendix F).

R is a contrast. R equals the squared norm of the output change caused by the changed expert selection plus a cross term with the existing deviation (Appendix B). The cross term can make R negative, so we read R only as a difference between the two runs; at 8 bits, its magnitude is at most 2.3% of R.

3.2 MEASURING AT TWO LEVELS

Layer-local and model-output measurements. We measure the deviation at two levels, under four quantization conditions (Table 2 in Appendix B): (i) the router weights of one layer, measured layer-locally (2–12 bits; Sections 4.2 and 4.3); (ii) the router weights of all layers and (iii) the

routed experts of all layers, measured at the model output (used only in the appendix); and (iv) both together, measured at the model output (2–10 bits; Section 4.1). Attention layers, embeddings, the output layer, shared experts, and router biases stay in full precision in all conditions. The layer-local measurement records the input to every MoE layer once, in a single forward pass of the original model. Then, for each layer in turn, it quantizes the router weights of that layer and recomputes only that layer from the recorded input. Because no deviation enters from earlier layers, it gives a separate value for each of the 191 MoE layers. The model-output measurement, used in conditions (ii)–(iv), quantizes the weights of all layers at once and measures the deviation in the final next-token distribution. It therefore also includes the routing changes that quantization error in earlier layers causes in later layers by altering their router input (Section 2.1).

Derived quantities and aggregation. Let T and $\hat{T}$ be the top-k sets that the original and quantized models select at layer ℓ. The change rate f is the fraction of tokens with $\hat{T} \neq T$ (Section 2.3); c is the mean of $|\hat{T} \setminus T|$ over tokens, that is, the number of changed experts per token; $h = R/f$ is the deviation per selection change; $g = R/c$ is the deviation per changed expert; and $\varepsilon = \|\hat{W} - W\|_F / \|W\|_F$ is the relative weight error. Quantities computed per layer (all quantities in the layer-local condition and f in the model-output conditions) are aggregated by the median across layers so that extreme layers do not dominate, and their variation is reported as the interquartile range across layers (Appendix F gives the results with the mean across layers).

3.3 Experimental Setup

Models. We use seven publicly released MoE models that fit on a single node and differ in router design (scoring function, router bias, and shared experts): Qwen1.5-MoE-A2.7B, OLMoE-1B-7B, DeepSeek-V2-Lite, Qwen3-30B-A3B, Moonlight-16B-A3B (sigmoid scores), GPT-OSS-20B (logit scores), and ERNIE-4.5-21B-A3B. Together they have 191 MoE layers (Appendix G lists the models and the checkpoint variants used).

Quantization. For the router, we quantize only the weight matrix W_g; for the experts, only the routed experts. All other weights stay in full precision. To exclude the effect of calibration data, all experiments use symmetric per-channel RTN at bit widths from 2 to 12, except the comparison of quantization methods in Section 4.2. That comparison quantizes the router weights of each layer (condition (i)) with both RTN and GPTQ on the same grid (group size 128, asymmetric), so that the two differ only in how they round (Section 2.2; GPTQ calibration: 128 sequences from the WikiText-2 training split; settings in Appendix G).

Data. We measure deviations on 64 sequences of 2,048 tokens drawn at random from the WikiText-2 test split (seed 1234), at 131,008 positions (2,047 per sequence). The MMLU-Pro experiments use a stratified sample of 2,000 questions from the test split, proportional to its 14 subject categories (seed 0).

Precision and statistics. The layer-local measurement loads the weights in bf16 and computes in fp32; the model-output measurement also computes in fp32 (Appendix G lists the exceptions for two models). Confidence intervals are 95% percentile intervals from a paired bootstrap with 1,000 resamples (seed 0), with sequences or questions as the resampling unit. We compute a ratio only when the interval of its denominator excludes zero.

4 Accuracy, the Reconstruction Error, and the Change Rate Miss the Routing-Induced Deviation

4.1 Accuracy Masks the Routing-Induced Deviation Because Gains and Losses Cancel

Setup. We quantize OLMoE, DeepSeek-V2-Lite, and Qwen3 to 4 and 8 bits under condition (iv) (experts + router; Section 3.2). The original model is the full-precision model before quantization. We give each of the 2,000 MMLU-Pro questions (Section 3.3) to the model once, zero-shot, together with its options. The answer is the option letter (A–J) with the highest probability at the

answer position. On MMLU-Pro, we measure two capability metrics (accuracy and the NLL of the correct option) and two distance metrics (answer disagreement with the original model and the KL divergence of the option distribution). On WikiText-2, we measure one capability metric (the NLL of the correct token) and two distance metrics (the output KL and the top-1 disagreement). Answer disagreement is the fraction of questions on which the answer differs from that of the original model; unlike the flips of Dutta et al. (2024), it also counts changes from one incorrect answer to another. Top-1 disagreement is the fraction of positions at which the most probable token differs from that of the original model. Here, we compute R with each of these distance metrics as D (Section 3). We say that a metric *detects* a change if the 95% interval of the mean per-question (MMLU-Pro) or per-sequence (WikiText-2) difference excludes zero (details in Appendix H).

Accuracy does not detect the answers that quantization changes. We quantize the whole model with RTN to 8 bits and compare it with the original model (free − original) on the 2,000 MMLU-Pro questions, for all three models (per-model table in Appendix H). In Qwen3 (original accuracy 0.458), 154 answers change, so the answer disagreement is 7.7%. But the net change in accuracy is 0.00 percentage points (pp) $[-0.70, +0.75]$. This is because 94 of these answers move from one incorrect answer to another, which accuracy by definition cannot detect. The other 60 split evenly into 30 correct-to-incorrect and 30 incorrect-to-correct changes, which cancel. The net change in accuracy is also indistinguishable from zero for OLMoE and DeepSeek (Appendix H). In OLMoE, from 4 bits, the bit width of the released checkpoints of Section 1, to 10 bits, the six settings change 3.7–53.7% of the answers, yet the net change in accuracy is indistinguishable from zero in all six (Appendix H). Accuracy therefore cannot tell whether PTQ achieves its design goal of keeping the outputs of the quantized model close to those of the original model (Section 1). Distance metrics compare the quantized model with the original model rather than with the correct answer, so, unlike accuracy, every departure from the original model adds to them instead of canceling.

Accuracy also misses what routing changes. Then, we restore only the expert selection of the same 8-bit RTN models to that of the original model (frozen − free), on MMLU-Pro and WikiText-2 for all three models. On MMLU-Pro, the change in accuracy is again indistinguishable from zero (-0.05 to $+0.40$ pp; all three intervals include zero). Yet R, the part of the deviation that routing causes (Section 3), is 47–68% of the answer disagreement with the original model and 89–93% of the option-distribution KL. The distance metrics show only the total deviation, and R shows how much of it comes from routing; accuracy shows neither. On WikiText-2, R is 85–89% of the output KL and 56–62% of the top-1 disagreement, yet the net change in the NLL of the correct token is indistinguishable from zero for all three models. Across tokens, the increases and decreases in this NLL almost cancel: the net change is 0.2–1.6% of the total change.

Even when accuracy detects a change, capability metrics disagree on its direction. With 4 bits added to the whole-model comparison (six settings), accuracy detects a change only in Qwen3 at 4 bits, where the option-distribution KL is largest (4.22, versus at most 0.41 in the other five); there, quantization lowers accuracy by 34.5 pp, but the NLL of the correct option reads the same model as 0.74 better (both intervals exclude zero).

Finding 1. Capability metrics such as accuracy and the NLL, which take the correct answer as the reference, are not reliable for detecting the changes that quantization makes; our metric R, measured with a distance metric, captures both the size and the share of the routing-induced deviation.

4.2 At High Bit Widths, the Reconstruction Error Decreases Faster than the Routing-Induced Deviation

Setup. We reduce the reconstruction error in two ways, by raising the bit width and by using calibration data as GPTQ does, and ask what happens to the routing-induced deviation in each case. Raising the bit width reduces the weight error ε, which measures how far the weights themselves are from their original values (Section 3.2). We quantize the router weights of the 191 layers of the seven models one layer at a time (condition (i)) and use the WikiText-2 test split (Section 3.3). For each layer, we measure the deviation that the reconstruction error captures (D_{frozen}) and the routing-induced deviation (R) at the layer-local level. Here, D_{frozen} and R are measured with the relative squared error of the MoE layer output (Eq. (4)). To raise the bit width, we use symmetric

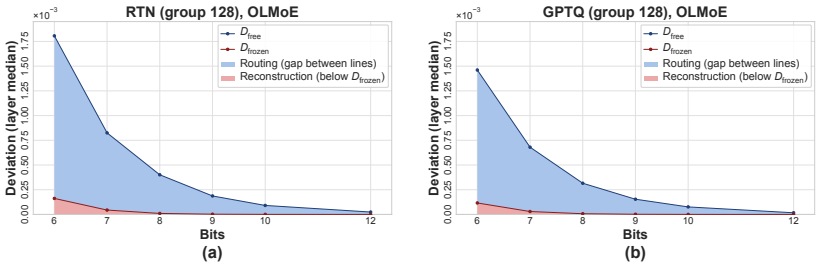

Figure 1: (a) RTN and (b) GPTQ on the same grid (group size 128, asymmetric) in OLMoE (condition (i), median across layers). As the bit width increases, both the deviation that the reconstruction error captures (D_{frozen}) and the routing-induced deviation (R, the gap between D_{free} and D_{frozen}) decrease, but D_{frozen} decreases faster.

per-channel RTN at 2 to 12 bits. To compare quantization methods, we quantize the same router weights with GPTQ and, as its reference, with RTN on the same grid (group size 128, asymmetric) at 3 to 12 bits. GPTQ uses as calibration data 128 sequences of 2,048 tokens drawn at random from the WikiText-2 training split. To find how the two deviations scale with the weight error, we use only high bit widths, $\{7, 8, 9, 10, 12\}$, where quantization is a small perturbation of the router scores (Appendix I gives the range of this high-bit regime and its rationale). For each layer, we regress $\log D_{\text{frozen}}$ and $\log R$ on $\log \varepsilon$ and call the slopes α_{frozen} and α_R. We aggregate by the median across layers.

The two deviations scale with different powers of the weight error. We raise the bit width with RTN. In all seven models, the deviation that the reconstruction error captures decreases as ε^2 and the routing-induced deviation as ε: $\alpha_{\text{frozen}} = 1.970$–$2.002$ and $\alpha_R = 0.982$–1.013 (95% intervals for each model in Appendix I). We fit both exponents in each of the 191 layers, and their Spearman correlation with layer depth satisfies $|\rho| \leq 0.28$ (Appendix I). With the selection fixed, the output is a smooth function of the weights, so its squared distance from the original output decreases as ε^2. In contrast, a token's selection changes only when quantization shifts its router scores by more than the gap between its k-th and $(k{+}1)$-th scores. Because this shift is proportional to ε, so is the change rate f (details in Appendix I).

The blind fraction follows a one-parameter function of the bit width. Then, from the same RTN runs, we obtain how the blind fraction changes with the bit width. Since $D_{\text{free}} = D_{\text{frozen}} + R$, it follows that $m/(1-m) = R/D_{\text{frozen}} \propto \varepsilon^{\alpha_R - \alpha_{\text{frozen}}} \approx \varepsilon^{-1}$. The weight error ε halves with every additional bit (the slope of $\log_2 \varepsilon$ per bit is -1.006 to -1.007; Appendix I), so the ratio doubles with every bit. In the high-bit regime, therefore,

$$\log_2 \frac{m}{1-m} = b - b^*. \tag{6}$$

We define b^*, the bit width at which the two deviations are equal ($m = 0.5$), as the *crossover bit width*; it differs across models (Appendix I). Fitted on some bit widths, Eq. (6) agrees with the measured m at the bit widths held out from the fit to within 0.015 in three models (Appendix I), and in all seven models m increases monotonically with the bit width in this regime (Appendix I). Even at 8 bits, where the change in accuracy in Section 4.1 was indistinguishable from zero, the blind fraction is 87.5–99.1% in the layer-local condition (Appendix I). The absolute deviations do not grow: both R and D_{frozen} decrease as the bit width increases, and what grows is the fraction of the deviation that the reconstruction error does not capture (Figure 1).

The same law holds with GPTQ. Finally, we compare GPTQ with RTN on the same grid, quantizing the same router weights with both. With RTN, the reconstruction error decreases only as the bit width increases, whereas GPTQ reduces it further at the same bit width. At 8 bits, GPTQ lowers the reconstruction error of the router weights (O_ℓ in Eq. (5)) to 47.5–68.3% of that of RTN in all seven models. In contrast, GPTQ increases the weight error ε by 9.9–15.7%. The two deviations follow the same powers as with RTN: $\alpha_{\text{frozen}} = 1.997$–$2.003$ and $\alpha_R = 0.992$–1.011. The relation

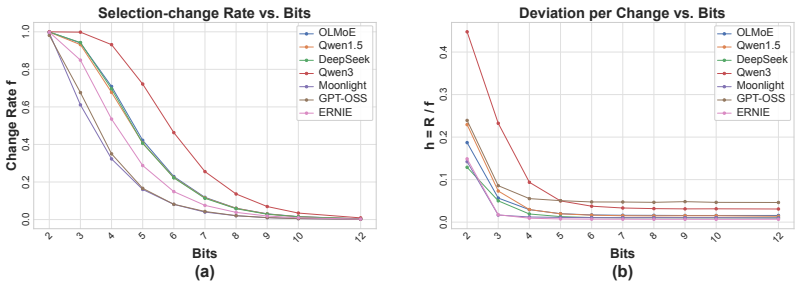

Figure 2: (a) Change rate f and (b) deviation per selection change $h = R/f$ versus bit width (condition (i), seven models, median across layers). At low bit widths, f saturates near 1, whereas h keeps increasing, so the routing-induced deviation $R = f \cdot h$ keeps growing after f stops rising.

$m/(1-m) \propto \varepsilon^{-1}$ behind Eq. (6) holds for both GPTQ and RTN. At 7 to 12 bits, the m of GPTQ is larger than or indistinguishable from that of RTN (Figure 1; Appendix I). GPTQ thus reduces the reconstruction error as designed but does not remove the routing-induced deviation.

Finding 2. As the bit width increases, the deviation that the reconstruction error captures decreases as the square of the weight error, but the routing-induced deviation only linearly, with RTN and GPTQ alike; at high bit widths, the reconstruction error alone misses most of the deviation.

4.3 The Change Rate Does Not Track the Size of the Routing-Induced Deviation

Setup. To capture the effect of routing, prior work uses routing-change metrics such as the change rate. However, the change rate reflects the size of the routing-induced deviation only if each selection change produces a deviation of similar size. We test this in the layer-local condition (i), as in Section 4.2: we quantize the router weights of the 191 layers of the seven models one layer at a time with symmetric per-channel RTN at 2 to 12 bits and use the WikiText-2 test split. For each layer, we measure the change rate f, the fraction of tokens for which at least one selected expert changes (Section 2.3), and the routing-induced deviation R. As in Section 4.2, R is measured with the relative squared error of the MoE layer output, and $h = R/f$ is the deviation per selection change (Section 3). All three are aggregated by the median across layers. By definition,

$$R = f \cdot h. \tag{7}$$

When the bit width decreases by one step, the increase in $\log R$ in each layer splits exactly into the increases in $\log f$ and $\log h$. We call $\Delta \log f / \Delta \log R$ the *share explained by the change rate*.

At high bit widths, the change rate tracks the deviation only within a model. At high bit widths (6 to 12 bits), where the expert selection changes rarely, almost all selection changes occur at the top-k boundary. At 8 bits, 88–99% of the experts that leave the selection were originally ranked k-th, and 86–97% of those that enter were originally ranked $(k{+}1)$-th (seven models). In this range, h barely changes within a model, so 97–100% of the increase in $\log R$ when the bit width decreases comes from the increase in $\log f$ (median across layers). Across models, however, h differs by a factor of about 7 at every bit width from 5 to 12 (6.99 [6.56, 7.40] between GPT-OSS and ERNIE at 8 bits; Figure 2b).

At low bit widths, the change rate stops rising while the deviation keeps growing. Lowering the bit width further breaks this relation (Figure 2a). The change rate counts only whether at least one of the selected experts changes, so once f is saturated near 1, it has nothing more to count: at 2 bits, f is 0.98–1.00 in all seven models. In the single step from 3 to 2 bits, f nearly stops rising (a factor of at most 1.6), whereas h increases by a factor of 1.9–8.6 (Figure 2b) and R by a factor of 2.0–14.1 across the seven models (intervals in Appendix J). In this step, the change rate explains only 0.3–21.9% of the increase in $\log R$ (at most 6.6% in five of the seven models), and the increase in h explains the rest (Appendix J). Counting changed experts instead of changed tokens gives the

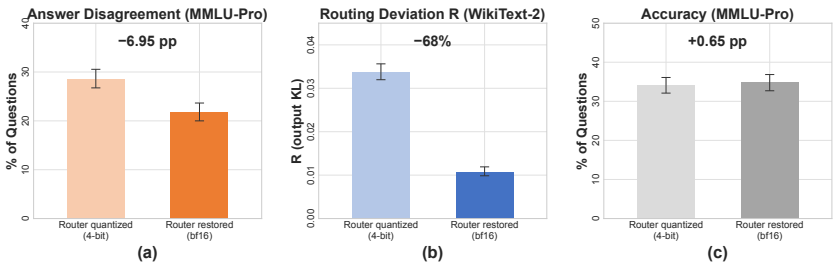

Figure 3: Restoring only the router weights of a released GPTQ checkpoint lowers (a) answer disagreement with the original model on MMLU-Pro. (b) The routing-induced deviation R (output KL on WikiText-2) shows this change, whereas (c) accuracy does not change measurably. Error bars: 95% intervals.

same result (Appendix J). At 2 bits, the experts that enter the selection also come from deep ranks: only 4.0–13.5% of them were originally ranked $(k{+}1)$-th (Appendix J). With f saturated, h still differs by a factor of 3.5 across models at 2 bits (Figure 2b), and so does $R = f \cdot h$. At low bit widths, the same change rate therefore does not mean the same routing-induced deviation, as at high bit widths, and obtaining R requires measuring h as well as f.

Finding 3. The change rate counts only how often the expert selection changes: it saturates at low bit widths while R keeps growing, and at high bit widths the same change rate corresponds to values of R about 7 times apart across models.

4.4 Experiment: Restoring the Router Weights of a Released Checkpoint

The frozen run of Section 4.1 forces the selection of the original model; it requires the original model and, by definition, removes R. To validate our findings, we restore only the router weights (0.0412% of all weights) of the community GPTQ checkpoint of Section 1 to their original bf16 values and keep all other quantized weights unchanged. This lets the model make its own expert selection, instead of fixing it to that of the original model. We measure R from the two runs, using the output KL on WikiText-2 as D (Section 3.3), and read each MMLU-Pro answer from the generated text (Appendix C). Restoring the router weights reduces answer disagreement with the original model by 6.95 pp [5.35, 8.55]; our metric R shows this change as a drop from 0.034 to 0.011 (a 68% decrease), whereas the change in accuracy is indistinguishable from zero ($+0.65$ pp $[-0.75, +1.95]$; Figure 3). Thus, to check whether a quantized model stays close to the original model, accuracy is not enough: it cannot detect the routing-induced deviation, but our metric R can.

5 Conclusion

Discussion. This paper shows why the routing-induced deviation of a quantized MoE model needs to be measured separately. The quantities currently used to evaluate quantized MoE models do not measure it. In capability metrics such as accuracy, gains and losses cancel. The deviation that the reconstruction error captures becomes small at high bit widths, but the routing-induced deviation remains. The expert-selection change rate saturates at low bit widths, and at high bit widths the same change rate corresponds to different deviations in different models. When evaluating a quantized MoE model, we propose measuring its routing-induced deviation with a distance metric. This measures both the size of the routing-induced deviation and its share of the total deviation. In a released checkpoint, restoring only the router weights lowers answer disagreement by 6.95 pp; our metric R shows this change as a 68% drop, whereas accuracy does not change measurably. We believe this will help improve PTQ methods for MoE models.

Limitations. We measure the routing-induced deviation but have not applied it to training; whether using it during training reduces the deviation remains untested.

AI USE STATEMENT

The research idea of this paper is the authors' own. We used generative AI tools only as assistants, for the purposes below; they did not write the paper or carry out its mathematics on their own.

Aid and polish writing. We used generative AI tools to check the English grammar, to make the text read naturally, and to help translate our drafts into English.

Retrieval and discovery. We asked for keywords to use when searching for related work. We also used the tools to check the publication venues of the cited papers and to format the references.

Research ideation and execution. We received help in deciding the concrete direction of the main experiments.

Drafting sections. We received advice on how to divide the experiments into sections and where to place each of them. The tools also helped draft parts of the text, which we then revised.

Mathematical claims. We derived the mathematical results by hand and used the tools to check the derivations once more.

Not used. We did not use generative AI tools to generate the research idea or synthetic datasets.

Responsibility. We have reviewed all AI-assisted work: we checked every statement about prior work against the original paper and every reported number against our experiment records. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

AutoAWQ. GitHub repository, 2025. URL `https://github.com/casper-hansen/AutoAWQ`.

Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. EAC-MoE: Expert-selection aware compressor for mixture-of-experts large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Lv Kaokao, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 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 *International Conference on Machine Learning*, 2026.

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. Accuracy is not all you need. In *Advances in Neural Information Processing Systems*, 2024.

Yi-Zeng Fang and Juinn-Dar Huang. Beyond freezing the router: Rank-aligned post-training quantization for mixture-of-experts models. *Transactions on Machine Learning Research*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

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

GPTQModel PR #1569. GitHub pull request, 2025. URL https://github.com/ModelCloud/GPTQModel/pull/1569.

GPTQModel PR #1995. GitHub pull request, 2025. URL https://github.com/ModelCloud/GPTQModel/pull/1995.

GPTQModel v5.0.0. GitHub release, 2025. URL https://github.com/ModelCloud/GPTQModel/releases/tag/v5.0.0.

Parvel Gu. Detecting a route flip is easier than knowing whether to fix it: Causal route-mediated damage in quantized mixture-of-experts. *arXiv preprint arXiv:2608.11212*, 2026.

Geonho Lee, Hancheol Park, Seunghyun Lee, Jungwook Choi, and Tae-Ho Kim. SRA-MoE: Output-aware selective router alignment for MoE quantization. In *ICML Workshop on Resource-Adaptive Foundation Model Inference*, 2026.

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. In *Conference on Machine Learning and Systems*, 2024.

llama.cpp. GitHub repository, 2026. URL https://github.com/ggml-org/llama.cpp.

mlx-lm. GitHub repository, 2026. URL https://github.com/ml-explore/mlx-lm.

Hancheol Park, Geonho Lee, Tairen Piao, and Tae-Ho Kim. Value-and-structure alignment for routing-consistent quantization of mixture-of-experts models. *arXiv preprint arXiv:2606.05688*, 2026a.

Hancheol Park, Geonho Lee, and Tae-Ho Kim. DREAM-MoE: Downstream routing error-aware margin-preserving quantization for mixture-of-experts large language models. In *ICML Workshop on Resource-Adaptive Foundation Model Inference*, 2026b.

Gokulakannan Sakthivel, Jerry Wu, Amogh Rajendra, and Giriprasad Radhakrishnan. Launch-bound and substitutable: Why three inference optimizations fail to pay off in mixture-of-experts models. *arXiv preprint arXiv:2608.26612*, 2026.

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

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

A Related Work

A.1 Routing in Mixture-of-Experts Models under Quantization

Sparse MoE layers send each token to a few experts chosen by a learned router (Shazeer et al., 2017; Fedus et al., 2022). Because the selection is a top-k over the router scores, a small perturbation of the scores can change the selected set. DREAM-MoE calls this quantization-induced change in expert selection *expert-shift*: small perturbations to router scores can make the quantized model execute a different expert set from the full-precision model (Park et al., 2026b). VSRAQ stresses that such rerouting does not merely add a continuous numerical error but changes the computation path itself (Park et al., 2026a). The router need not be quantized for this to happen: quantization errors in the expert outputs shift the logits of the next router, so a router kept in full precision can still select different experts (Fang & Huang, 2026). Sakthivel et al. (2026) find that the experts that change are overwhelmingly those ranked 8th and 9th, where the logit margin is thinnest.

A.2 Post-Training Quantization of MoE Models

Layer-wise and block-wise reconstruction. GPTQ and AWQ quantize one linear layer at a time and choose the quantized weights so that the output of that layer on calibration data stays close to the original (Frantar et al., 2023; Lin et al., 2024). Block-wise methods reduce the output error of a whole decoder block instead (Shao et al., 2024; Cheng et al., 2024). VSRAQ points out that minimizing only the reconstruction loss can lead to solutions in which the block outputs are numerically close but the selected experts have changed (Park et al., 2026a).

Keeping or quantizing the router. Quantization toolkits locate the linear layers of a model by name or module type, quantize them, and keep in full precision only the layers on an exclusion list. Router weights implemented as a linear layer are quantized along with the other weights unless they appear on this list. With the same toolkit, the router stays unquantized in models that do not implement it as a linear-layer module (Sakthivel et al., 2026). RouteQuant observes that prior methods either keep the router in full precision or quantize it, depending on the system design (Fang & Huang, 2026). EAC-MoE and SRA-MoE keep the router in full precision (Chen et al., 2025; Lee et al., 2026), whereas EAQuant and RouteQuant quantize it to 8 bits (Fu et al., 2025; Fang & Huang, 2026). Sakthivel et al. (2026) report that leaving the routers in FP16 lowers the routing drift but raises the loss.

Routing-aware calibration. Several methods add a routing term to the calibration objective. EAC-MoE calibrates the router of each layer with a squared error over the top-K router logits (Chen et al., 2025). EAQuant aligns the router logits with a squared error and a KL divergence restricted to the routing-relevant experts (Fu et al., 2025). VSRAQ adds a term that preserves the order of the top-k scores and the margin at the top-k boundary (Park et al., 2026a). DREAM-MoE penalizes violations of the score margins between the selected and near-boundary experts, both in the current block and in the router of the next block (Park et al., 2026b). RouteQuant combines a rank-weighted Jaccard loss with a hinge on the gaps between consecutive top-k scores (Fang & Huang, 2026). SRA-MoE aligns the routing only for the tokens whose routing shifts change the output (Lee et al., 2026). All of these terms act during calibration to keep the original expert selection.

A.3 Measuring Routing Change and Evaluating Quantized Models

Table 1 lists how prior work measures routing change (Section 2.3).

Table 1: How prior work measures routing change.

Work	What is compared for each token	Aggregation
EAC-MoE (Chen et al., 2025)	Whether all / at least one / at least half of the selected experts change (three criteria)	Per layer, averaged over layers
GEMQ (Deng et al., 2026)	Whether the token is routed to a different expert	Per layer, averaged over layers
VSRAQ (Park et al., 2026a)	Jaccard similarity of the two top-k sets	Per layer
DREAM-MoE (Park et al., 2026b)	Fraction of experts shared by the two top-k sets	Per block, on a 32-example probe set
RouteQuant (Fang & Huang, 2026)	Rank, in the quantized model, of each expert selected by the original model (lower score for larger rank shifts)	Averaged over layers, ranks, and tokens
Sakthivel et al. (2026)	Jaccard distance of the two top-k sets; number of changed experts	One value per model, over all layers and token positions
SRA-MoE (Lee et al., 2026)	Complement of rank-biased overlap (RBO)	Distribution over tokens
Dutta et al. (2024)	*(Not routing; per question)* Whether the answer changes between correct and incorrect	Per task
Ours	Whether at least one selected expert changes	Per layer, median over layers

Capability and distance metrics. Dutta et al. (2024) distinguish capability metrics, such as accuracy and perplexity, from distance metrics, such as the KL divergence and flips. On dense models, they observe that even when the accuracy of the original and compressed models is similar, answers change from correct to incorrect and vice versa in proportion, and they describe flips as a warning that the compressed model behaves differently. For MoE models, Sakthivel et al. (2026) report quality as the NLL of the correct answer rather than as task accuracy, because accuracy at a feasible evaluation budget cannot resolve the differences involved.

Restoring the original routing. Prior work also restores routing decisions to see how much they matter. RouteQuant runs the quantized model with the top-k experts of the full-precision router at every token and measures the change in accuracy (Fang & Huang, 2026). Sakthivel et al. (2026) replay the routes of the quantized model through full-precision weights and find that they reproduce only 2.7% of the quality loss. SRA-MoE replaces the routing decisions of a subset of tokens with those of the original model and observes that the output discrepancy is often reduced (Lee et al., 2026). These experiments use one or two bit widths; our frozen run fixes the expert selection of every token to measure R with a distance metric across bit widths (Section 3; Appendix E.5 compares the procedures).

B Measurement Details

B.1 R Is a Contrast

Let $e_{\text{frozen}} = \hat{y}^{\text{frozen}} - y$ and $r = \hat{y}^{\text{free}} - \hat{y}^{\text{frozen}}$. When D is a squared distance, omitting the common denominator,

$$R = \|e_{\text{frozen}} + r\|^2 - \|e_{\text{frozen}}\|^2 = \|r\|^2 + 2\langle e_{\text{frozen}},\, r\rangle. \tag{8}$$

Equation (8) shows that R is the squared norm of the output change caused by the changed expert selection, $\|r\|^2$, plus a cross term. The cross term is the inner product of the existing deviation e_{frozen} and the change r, so it shows how r is aligned with e_{frozen}; it is positive or negative depending on their directions. Hence R can be negative, and we read it only as a difference between the two runs. At high bit widths, $R \approx \|r\|^2$: at 8 bits, the magnitude of the cross term is at most 2.3% of R (Appendix E).

B.2 Quantization Conditions

Table 2 lists the four quantization conditions of Section 3.2.

Table 2: Quantization conditions. Attention layers, embeddings, the output layer, shared experts, and router biases stay in full precision in all conditions.

Condition	Measured at	Quantized weights	Bits	Section
(i) Router (one layer)	Layer-local	Router weights of one layer	2–12	4.2, 4.3
(ii) Router	Model output	Router weights of all layers	3–12	Appendix
(iii) Experts	Model output	Routed experts of all layers	4–8	Appendix
(iv) Experts + router	Model output	(ii) and (iii) together	2–10	4.1

C Released Checkpoints and Quantization Toolkits

This appendix gives the details behind the two released checkpoints of Section 1 and the experiment of Section 4.4: how the two checkpoints differ (C.1), how we restore the router weights (C.2), the two-run measurement on WikiText-2 (C.3), how we read the answers on MMLU-Pro (C.4), and the audit of quantization toolkits behind Section 1 (C.5).

C.1 The two released checkpoints

Table 3 compares `JunHowie/Qwen3-30B-A3B-GPTQ-Int4` (the community checkpoint) and `Qwen/Qwen3-30B-A3B-GPTQ-Int4` (the official checkpoint). We read their configuration files

and the headers of their weight files (tensor names, data types, and shapes). Both use 4 bits, group size 128, symmetric quantization, no act-order (`desc_act` = false), and the GPTQ checkpoint format, and both have 30,532,122,624 parameters. They quantize the same modules except one: the community checkpoint stores the router of each of the 48 MoE layers (`mlp.gate`) as packed 4-bit integers with F16 scales and group indices, whereas the official checkpoint stores it unquantized in F16; these values differ from the original bf16 router weights only by fp16 rounding, in 3,823 of their elements. The router weights hold 12,582,912 parameters (48 layers $\times$ 128 experts $\times$ 2,048 inputs), or 0.0412% of all parameters, and this number is exactly the difference between the quantized parameter counts of the two checkpoints (97.961% versus 97.920% of all parameters).

The two checkpoints share these settings but not their quantized values; they come from different GPTQ runs. Of the 18,624 quantized attention and expert tensors that both contain, none is bit-identical between the two. Over these tensors, the relative distance between the two dequantized weight sets, $\|\hat{W}_{\text{off}} - \hat{W}_{\text{com}}\|_F / \|W\|_F$, is 0.1521, the cosine between their quantization errors is 0.330, and 29.9% of the dequantized elements are equal. The configuration files also differ outside the settings above (Table 3). Because the two checkpoints differ in more than the router, Section 4.4 does not compare them with each other. It compares the community checkpoint with the same checkpoint after restoring only its router weights (Appendix C.2), and the official checkpoint appears only as a reference row in Tables 4 and 5. The configuration of the community checkpoint names GPTQModel 4.0.0 as the toolkit that produced it, and its current weight files were uploaded to the repository on 2025-09-05. GPTQModel 4.0.0 quantizes the Qwen3-MoE router by default (Appendix C.5).

Table 3: The two released checkpoints (from configuration files and weight headers).

	Community (JunHowie)	Official (Qwen)
Bits / group size / symmetric	4 / 128 / true	4 / 128 / true
Act-order (`desc_act`) / format	false / GPTQ	false / GPTQ
Router (`mlp.gate`), 48 layers	4-bit (packed I32, F16 scales)	F16, unquantized
Attention and routed experts	4-bit	4-bit
Quantized parameters	97.961%	97.920%
Embeddings, norms, and output layer	BF16	F16
Toolkit recorded in the configuration	GPTQModel 4.0.0	none
Dampening	not recorded	0.01
Calibration data	not recorded	not recorded

C.2 Restoring the router weights

The checkpoint with restored router weights differs from the community checkpoint only in the router. In each of the 48 MoE layers, the four tensors that store the quantized router (`mlp.gate.qweight`, `.scales`, `.qzeros`, and `.g_idx`) are replaced by the router weight `mlp.gate.weight` of the original bf16 model; no other tensor changes, and nothing is quantized again. In our runs, we dequantize the community checkpoint and overwrite its 48 router weights with those of the original model, and the restored router weights are bit-identical to the original ones. All three checkpoints are dequantized and run with the same Transformers code as the original model, not with 4-bit kernels (fp16 on WikiText-2 and bf16 for generation; Appendix G).

C.3 Two-run measurement on WikiText-2

Table 4 reports the measurement behind Section 1 and Section 4.4. Each checkpoint runs as released, with its attention and expert weights quantized (Table 3). As in conditions (ii)–(iv) of Table 2, we measure its deviation at the model output, here on the 131,008 WikiText-2 positions of Section 3.3 in fp16. D is the output KL divergence, and the frozen run forces the top-k indices of the original model at every layer (Section 3.1). The change rate f is computed for each layer as the fraction of positions whose top-k set differs from that of the original model; the table reports its median and mean across the 48 layers. Intervals are for the difference between the community checkpoint and the restored checkpoint (paired bootstrap over the 64 sequences; Appendix F.3).

Table 4: Two-run measurement on WikiText-2 (fp16). D: output KL.

	Original	Community (router 4-bit)	Router restored	Official (ref.)	Community − restored [95%]
Change rate f (layer median)	0	0.7589	0.4316	0.4395	$+0.327$ [0.323, 0.330]
Change rate f (layer mean)	0	0.7277	0.4284	0.435	$+0.299$ [0.297, 0.302]
Changed experts per token	0	1.043	0.4894	0.5017	$+0.554$ [0.542, 0.564]
Top-1 agreement	1	0.8769	0.9031	0.9020	-0.0262 [-0.0281, -0.0244]
D_{free}	0	0.1024	0.06035	0.06288	$+0.0421$ [0.0400, 0.0443]
D_{frozen}	0	0.06871	0.04953	0.05228	$+0.0192$ [0.0182, 0.0204]
R	0	0.03372	0.01082	0.01061	$+0.0229$ [0.0213, 0.0245]
m	–	0.3292	0.1792	0.1687	$+0.150$ [0.136, 0.163]

Restoring the router weights lowers R from 0.0337 to 0.0108 (a 68% decrease) and D_{frozen} from 0.0687 to 0.0495. Of the decrease in D_{free} (0.0421), R makes up 0.0229 (54%) and D_{frozen} the other 0.0192 (46%). In the frozen runs, the two checkpoints use the same expert selection, so the decrease in D_{frozen} comes from the gate values, the only part of the computation that differs between these two runs. The change rate falls from 75.9% to 43.2% (median across layers; 72.8% to 42.8% as the mean). The official checkpoint, whose router weights equal those of the original model in this fp16 measurement, still sends 43.9% of tokens to a different expert set: the quantized weights before each router alter its input (Section 2.1). Figure 4 shows the change rate of each layer.

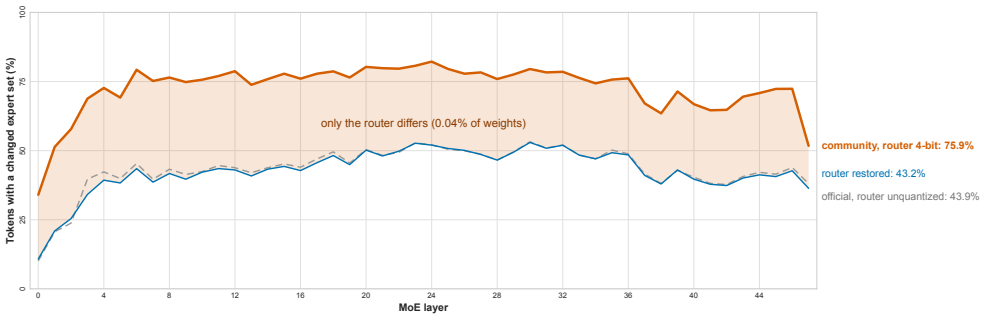

Figure 4: Change rate of each MoE layer (WikiText-2).

C.4 Answers on MMLU-Pro

Section 4.4 reads each answer from generated text. We give each of the 2,000 MMLU-Pro questions of Section 3.3 to the model once, in the chat template of Qwen3 with thinking disabled, as the question and its lettered options followed by the line "Answer with the letter only." We decode greedily for at most 16 new tokens, with the same batches (16 questions, left padding, same order) for the original model and the three checkpoints, and compute in bf16. The answer is the earliest standalone capital letter from A to J in the decoded text; if there is none, the answer is unparsed. Accuracy counts an unparsed answer as incorrect. Answer agreement counts two answers as equal when they are the same letter or both unparsed. Intervals are paired bootstrap intervals over questions (1,000 resamples, seed 0). Running the original model twice gives identical output tokens (checked on one batch of 16 questions).

This procedure differs from the option-letter scoring of Section 4.1, which reads the answer from the option-letter probabilities at the answer position (Appendix H.1). The original model scores 0.458 under that scoring and 0.408 under this one, where 26.1% of its answers are unparsed and count as incorrect. Within each procedure, every quantized model is compared with the original model on the same questions.

Restoring the router weights raises answer agreement by 6.95 percentage points [5.35, 8.55], whereas the change in accuracy, $+0.65$ points [-0.75, $+1.95$], is indistinguishable from zero. The restored checkpoint changes 139 fewer answers than the community checkpoint (434 versus 573),

Table 5: MMLU-Pro answers read from generated text. Agreement: same answer as the original model. C→I, I→C: correct-to-incorrect and incorrect-to-correct. Unparsed: no answer letter. Brackets: 95% intervals.

Checkpoint	Accuracy	Agreement	Changed	C→I	I→C	Unparsed
Original	0.4080 [0.387, 0.429]	–	–	–	–	26.1%
Community	0.3415 [0.321, 0.361]	0.7135 [0.695, 0.733]	573	201	68	36.2%
Router restored	0.3480 [0.327, 0.369]	0.7830 [0.764, 0.800]	434	157	37	36.0%
Official (ref.)	0.3845 [0.364, 0.405]	0.8245 [0.808, 0.841]	351	102	55	29.7%

and its correct-to-incorrect and incorrect-to-correct changes fall together (157 and 37, versus 201 and 68).

Unparsed answers do not produce the difference. About a third of the answers of the quantized checkpoints are unparsed, and two unparsed answers count as agreeing. Table 6 recomputes the difference in agreement in three other ways from the same generations: using only the questions on which the original model's answer is parsed, using only the questions on which the answers of the original model and of both compared checkpoints are parsed, and counting an unparsed answer as disagreeing even when both answers are unparsed. The difference stays positive, with intervals that exclude zero, in all three (5.85 to 7.92 points). The parser accepts a letter beyond the number of options of a question in at most one of the 2,000 questions for each model (353 questions have fewer than 10 options).

Table 6: Agreement gain from restoring the router weights (percentage points), by how unparsed answers are handled.

Unparsed answers	Questions	Gain [95%]
Two unparsed answers agree (Section 4.4)	2,000	+6.95 [5.35, 8.55]
Only questions with a parsed original answer	1,478	+7.92 [5.95, 9.95]
Only questions with all three answers parsed	1,193	+6.79 [4.53, 8.97]
Unparsed answers count as disagreeing	2,000	+5.85 [4.35, 7.35]

C.5 Quantization toolkits

Section 1 states that toolkits do not agree on whether to quantize the router. Table 7 shows the default behavior of nine toolkit versions (eight toolkits, with GPTQModel at two versions) for seven MoE architectures. We read the source code of each without installing or running it, at fixed commits (GPTQModel 4.0.0 `40759cdf` and 7.5.0 `7490b66b`, llm-compressor `de46bfd5`, AutoRound `0439590b`, AutoAWQ `88e4c76b`, Transformers 5.17 `856157a2`, llama.cpp `1af554f8`, mlx-lm `ed1fca4c`, HQQ `835e2590`). For each, we followed how it selects the modules to quantize, whether its default exclusion list contains the router, and whether a model-specific rule overrides either.

Three of the nine versions quantize the Qwen3-MoE router by default (GPTQModel 4.0.0, AutoAWQ 0.2.9, and mlx-lm 0.31.3 at 8 bits), and a fourth does so when HQQ runs standalone with Transformers 4.x. Five of the nine quantize the router of at least one of the seven architectures. The rules that decide this differ in kind:

- GPTQModel 4.0.0 lists `mlp.gate` among the modules to quantize for Qwen3-MoE and Qwen2-MoE (`qwen3_moe.py`, line 35); GPTQModel 7.5.0 marks the router as excluded (`gate:!`, `router:!`).
- AutoAWQ quantizes every `nn.Linear` module inside a decoder block, with an exception only for Mixtral (`mixtral.py`, line 20).
- AutoRound excludes every module whose name ends in `.gate` in MoE models (`special_model_handler.py`, lines 1171–1174); its GGUF output path turns this rule off, and we did not follow that path.

Table 7: Does each toolkit quantize the router by default? Y: yes (default bit width, where known); N: no; –: architecture not supported; TF: Transformers. §With TF 4.x. ‡Checked up to module selection, not run.

Toolkit	Qwen3	Qwen2	Mixtral	OLMoE	Phi-3.5	DSV2	gpt-oss
GPTQModel 4.0.0	Y (4)	Y	N	–	N	N	–
GPTQModel 7.5.0	N	N	N	–	N	N	N
llm-compressor 0.13.0	N	N	N	N	Y‡	N	N
AutoRound 0.15.1	N	N	N	N	N	N	N
AutoAWQ 0.2.9	Y§	–	N	–	–	N	–
bitsandbytes (TF 5.17)	N	N	N	N	N	N	N
llama.cpp b11053	N	N	N	N	N	N	N
mlx-lm 0.31.3	Y (8)	Y (4)	Y (4)	Y (4)	Y (4)	N	Y (8)
HQQ 0.2.8 (TF 5.17)	N	N	N	N	Y‡	N	N
HQQ 0.2.8 (standalone, TF 4.x)	Y	Y	Y	Y	Y	N	N

- llama.cpp excludes the router tensor (`ffn_gate_inp.weight`) by name and stores it in F32 (`llama-quant.cpp`, line 306); its `--tensor-type` option cannot re-enable it.
- mlx-lm quantizes every module whose input dimension divides the group size, unless a model-specific predicate says otherwise; the predicates of Qwen3-MoE and gpt-oss set the router to 8 bits.
- Whether bitsandbytes and HQQ quantize the router depends on the Transformers version. In Transformers 4.57.1, the routers of Qwen3-MoE, Qwen2-MoE, Mixtral, OLMoE, and Phi-3.5-MoE are `nn.Linear` modules, and both toolkits quantize them. In Transformers 5.14 and 5.17, no router is an `nn.Linear` module except that of Phi-3.5-MoE, which subclasses it; toolkits that select modules by subclass (HQQ, llm-compressor) quantize it, and bitsandbytes, which requires the exact class `nn.Linear`, does not.

The same toolkit changed its default. GPTQModel added the Qwen3-MoE router to its quantization targets in pull request #1569 (commit `48d2452`, merged 2025-04-30) and removed it in pull request #1995 (commit `ad82cd6`, merged 2025-10-07), with the inline comment "0.5MB per layer. Not worth quantizing". The released versions 4.0.0 (2025-08-23), 4.1.0, 4.2.0, and 4.2.5 (2025-09-16) therefore quantize the router of Qwen3-MoE by default, and no other version was released between 4.2.5 and 5.0.0 (2025-10-24). Version 5.0.0 excludes the router for GPTQ but keeps it for AWQ through a method-specific override (`module_tree_overrides`), so within one version the choice depends on the quantization method. In the pull requests and commits we examined, neither change cites a measurement: #1569 has no description, review, or discussion, and its commit message reads "fix hook_module and qwen3_moe"; #1995, titled "Bug fixes", gives only the inline comment above as a reason.

D Router Quantization in Checkpoints on the Hugging Face Hub

Section 1 states that the released quantized variants of MoE models do not agree on whether to quantize the router. This appendix describes how we surveyed 300 such variants (D.1), how many quantize the router (D.2), and what their model cards and configuration files state about it (D.3).

D.1 Collection

We collected quantized derivatives of ten MoE base models (Table 8) on 2026-09-19. For each base, we listed the repositories that the Hub tags as quantized derivatives of that base, sorted by downloads, and added repositories found by searching model names; this gave 1,295 candidates. From these, we selected 300 repositories, taking them from each base in turn in order of downloads. The 300 repositories contain 2,274 weight files, counting each quantization variant of a GGUF repository separately.

For each base, we used the name of its router tensor, for example `mlp.gate.weight` for Qwen3, `block_sparse_moe.gate.weight` for Mixtral, `mlp.router.weight` for gpt-oss,

and `blk.i.ffn_gate_inp.weight` in GGUF files. We then read only file headers: the safetensors header of the shard that holds the router, and the leading 8–64 MB of each GGUF file. A file quantizes the router if the router tensor is stored in a quantized format (packed integers with scales, an 8-bit floating-point type, or a GGUF quantization type) rather than as a 16- or 32-bit floating-point tensor. A repository quantizes the router if at least one of its files does (39 repositories; in 37 of them, every file with a readable status does). Our reader classifies the two checkpoints of Appendix C correctly: the community checkpoint as quantized and the official one as unquantized. The status of 24 repositories is unreadable, mostly GGUF files without an MoE router tensor (for example, draft models and single-expert extractions) and formats without a tensor table.

D.2 How many variants quantize the router

Of the 276 repositories whose status we determined, 39 quantize the router (0.141 [0.105, 0.187]). The fraction depends on the base model (Table 8) and much more on the file format (Table 9): 24 of the 25 MLX repositories quantize the router (the exception is a bf16 conversion), whereas 155 of the 158 GGUF repositories keep it in full precision, as llama.cpp does by default (Appendix C.5). Weighted by downloads over the past 30 days, the fraction is 0.998 for MLX and 0.002 for GGUF. Intervals for counts are Wilson score intervals (95%, no continuity correction); intervals for download-weighted fractions come from a bootstrap over repositories (1,000 resamples, seed 0).

Table 8: Variants that quantize the router, by base model. Read: router status readable. Quant.: router quantized. Downloads: share of 30-day downloads. Brackets: 95% intervals.

Base model	Repos	Read	Quant.	Fraction	Downloads
Qwen3-30B-A3B	34	34	6	0.176 [0.083, 0.335]	0.039 [0.006, 0.122]
Qwen3-235B-A22B	34	31	8	0.258 [0.137, 0.432]	0.064 [0.008, 0.230]
Mixtral-8x7B	33	23	2	0.087 [0.024, 0.268]	0.072 [0.000, 0.272]
Mixtral-8x22B	28	25	5	0.200 [0.089, 0.391]	0.001 [0.000, 0.044]
DeepSeek-V2-Lite	33	33	0	0.000 [0.000, 0.104]	0.000
Qwen1.5-MoE-A2.7B	33	33	3	0.091 [0.031, 0.236]	0.028 [0.000, 0.086]
gpt-oss-20b	33	31	4	0.129 [0.051, 0.289]	0.217 [0.002, 0.654]
gpt-oss-120b	33	27	5	0.185 [0.082, 0.367]	0.118 [0.004, 0.463]
Phi-3.5-MoE	23	23	2	0.087 [0.024, 0.268]	0.041 [0.000, 0.392]
OLMoE-1B-7B	16	16	4	0.250 [0.102, 0.495]	0.018 [0.002, 0.074]
All	300	276	39	0.141 [0.105, 0.187]	0.127 [0.019, 0.322]

Table 9: Variants that quantize the router, by file format (columns as in Table 8).

Format	Repos	Read	Quant.	Fraction	Downloads
GGUF	177	158	3	0.019 [0.006, 0.054]	0.002 [0.000, 0.006]
MLX	25	25	24	0.960 [0.805, 0.993]	0.998 [0.975, 1.000]
bitsandbytes	10	10	7	0.700 [0.397, 0.892]	0.100 [0.001, 0.998]
GPTQ	12	12	2	0.167 [0.047, 0.448]	0.000 [0.000, 0.034]
compressed-tensors	28	28	2	0.071 [0.020, 0.226]	0.015 [0.000, 0.188]
safetensors, other	7	7	1	0.143 [0.026, 0.513]	0.002 [0.000, 0.017]
AWQ	10	10	0	0.000 [0.000, 0.278]	0.000
MXFP4	9	9	0	0.000 [0.000, 0.299]	0.000
AutoRound	6	6	0	0.000 [0.000, 0.390]	0.000
FP8	4	4	0	0.000 [0.000, 0.490]	0.000
Quark	3	3	0	0.000 [0.000, 0.561]	0.000
ModelOpt	3	3	0	0.000 [0.000, 0.561]	0.000
GGUF and safetensors	1	1	0	0.000 [0.000, 0.793]	0.000
No tensor table	5	0	–	–	–

The most downloaded repository that quantizes the router is an MLX conversion of gpt-oss-20b (274,651 downloads in 30 days, 70% of the downloads of the 39 repositories).

Checkpoints produced while GPTQModel quantized the router. Appendix C.5 shows that the main branch of GPTQModel included the Qwen3-MoE router in its quantization targets between two pull requests (2025-04-30 to 2025-10-07). Table 10 lists the Qwen3 GPTQ repositories created in this window; six of the eight are outside the 300; we took them from the same candidate list and read them in the same way. The five whose configuration names GPTQModel as the producing toolkit all quantize the router, and the other three, which do not name GPTQModel, all keep it in full precision. The four Qwen3 GPTQ repositories created after the window all keep the router in full precision.

Table 10: Qwen3 GPTQ repositories created while the main branch of GPTQModel quantized the router (between PRs #1569 and #1995). Toolkit: as recorded in the configuration. Router: quantized or kept in full precision. Downloads: past 30 days.

Repository	Created	Toolkit	Bits	Router	Downloads
JunHowie/Qwen3-30B-A3B-GPTQ-Int4	2025-04-30	GPTQModel 4.0.0	4	quantized	26
NePe/Qwen3-30B-A3B-GPTQ	2025-05-01	GPTQModel 3.1.0-dev	4	quantized	159
btbtyler09/Qwen3-30B-A3B-gptq-8bit	2025-05-02	GPTQModel 3.1.0-dev	8	quantized	14
Qwen/Qwen3-30B-A3B-GPTQ-Int4	2025-05-05	–	4	kept	187,046
Qwen/Qwen3-235B-A22B-GPTQ-Int4	2025-05-10	–	4	kept	1,372
JunHowie/Qwen3-30B-A3B-GPTQ-Int8	2025-05-13	GPTQModel 4.0.0	8	quantized	61
QuantTrio/Qwen3-235B-A22B-GPTQ-Int8	2025-05-14	–	8	kept	34
btbtyler09/Qwen3-30B-A3B-gptq-4bit	2025-07-05	GPTQModel 4.0.0-dev	4	quantized	17

D.3 What model cards and configuration files state

We also searched the model cards and configuration files of the same repositories for statements that the router is kept out of quantization. We count a statement when a word for exclusion or keeping (for example *ignore*, *exclude*, *not_convert*, *skip*, *keep*, *fp16*, or *bf16*) and a name of the router (*router*, *gate*, `mlp.gate`, or `block_sparse_moe.gate`) appear on the same line. We have the card of 277 repositories (its leading 200 lines) and a configuration file of 120.

- Of the 277 cards, 11 (4.0%) state that the router is kept out of quantization. Six of these 11 derive from gpt-oss, whose released checkpoint already stores only the experts in MXFP4; without them, the fraction is 5 of 244 (2.0%).
- Of the 120 configuration files, 45 (37.5%) list the router among the modules excluded from quantization.
- Among the 102 repositories with both, the configuration excludes the router in 39, and the card of 32 of these 39 (82%) does not mention it.
- The keyword `modules_to_not_convert` appears in 0 of the 277 cards and in 24 of the 120 configuration files.

The official FP8 checkpoints of Qwen3 (`Qwen/Qwen3-30B-A3B-FP8` and `Qwen/Qwen3-235B-A22B-FP8`) are examples: their configuration lists `mlp.gate` under `modules_to_not_convert`, and their cards do not mention it. To tell whether a released checkpoint quantizes its router, its files are therefore a more reliable source than its model card.

Limitations of the survey. We read only the leading 200 lines of each card, so we can have missed statements further down; this can only lower the counts of cards that state the choice. We did not filter the repositories by whether they are quantized, so the 300 include re-uploads in 16-bit precision, merged models, and GGUF conversions. The three denominators (277 cards, 120 configuration files, 102 with both) differ and are reported separately.

E Validity of the Free and Frozen Runs

Section 3.1 defines R as the difference between the deviations of the free and the frozen run. This appendix checks that the forcing itself changes nothing but the expert selection (E.1), reports what

changes when the gate values are fixed as well (E.2), measures the cross term of Eq. (8) (E.3), lists the settings in which R is negative (E.4), and compares the procedure with prior experiments that restore the original expert selection (E.5).

E.1 The forcing introduces no deviation

The frozen run replaces only the top-k indices that the quantized router selects with the indices that the original model selects at the same position and layer. The gate values are computed from the scores of the quantized router with the model's own gating function, restricted to the forced indices. The model-output experiments check up to three properties of this procedure:

1. running the original model twice gives bit-identical outputs;
2. running the original model while forcing its own top-k indices gives outputs bit-identical to those of the unforced run;
3. in the frozen run of each quantized model, the top-k indices at every position and layer equal those of the original model.

The layer-local experiment instead checks that recomputing a layer from its recorded input reproduces the output of the forward pass bit for bit. Table 11 lists the checks of each experiment; all of them passed. The measurement floor, the deviation between two runs that should be identical, is therefore exactly zero, and the only difference between the free and the frozen run of a quantized model is its expert selection.

Table 11: Checks of the forcing, by experiment (all passed).

Experiment	Checks
Layer-local, condition (i) (Sections 4.2, 4.3)	Recomputed layer = forward-pass output, bit for bit (3 layers per model). Original weights unchanged throughout the run (all layers).
Model output, conditions (ii)–(iv)	Original model twice: identical. Original indices forced: identical. Weights restored after each setting: identical.
Section 4.1	Original model twice: identical logits and indices. Frozen runs: zero selection changes.
Section 4.4	Original model twice: identical. Original indices forced: identical. Frozen runs: original selection.
Appendix E.2	Original model twice: identical logits, indices, and gate values. Indices forced, and indices and gate values forced: identical.
Appendix K	As for the model output; frozen runs: original selection (13/13 and 16/16 settings).

E.2 Fixing the gate values as well

The router determines both the expert selection and the gate values (Section 2.1), and the frozen run fixes only the selection. To see how much of the deviation the gate values carry, we ran a third run that replaces the whole output of each router, both the indices and the gate values, with that of the original model. We ran it on Qwen3-30B-A3B in condition (iv) at 8, 9, and 10 bits, in fp16 on 16 sequences of 2,048 tokens from the start of the WikiText-2 test split, with D the output KL divergence (Table 12).

Fixing the gate values as well removes 95.5% of the deviation at 8 bits, compared with 87.5% when only the indices are fixed; the difference between the two runs is -4.18×10^{-4} [-4.73×10^{-4}, -3.61×10^{-4}] at 8 bits, and its interval excludes zero at 9 and 10 bits as well (paired bootstrap over the 16 sequences). Of the deviation that remains after the indices are fixed, the gate values carry 63.8% at 8 bits and 49.8% at 10 bits. We leave this part in D_{frozen} because the gate values vary continuously with the router scores, whose error the reconstruction error of the router measures (Section 3.1); only the discrete change in expert selection is outside it. Replacing the whole router output also removes the effect that quantized earlier layers have on the router input (Section 2.1), so this run is a diagnostic, not a setting that keeping the router in full precision would reproduce. On the 64 sequences of Section 3.3, fixing only the indices removes 88.6% [87.5, 89.7] of the output KL of the same model at 8 bits (Appendix I.3).

Table 12: Fixing the indices only versus the indices and gate values (Qwen3, condition (iv), 16 sequences). D in units of 10^{-4}. Share removed $= 1 - D/D_{\text{free}}$. Last column: share of D_{frozen} removed by also fixing the gate values.

			Indices only		Indices and gates		
Bits	f	D_{free}	D_{frozen}	Removed	D	Removed	Gate share
8	0.2170	52.23	6.551	0.875	2.374	0.955	0.638
9	0.1433	39.63	1.748	0.956	0.736	0.981	0.579
10	0.1105	20.66	0.567	0.973	0.285	0.986	0.498

E.3 The cross term of Eq. (8)

Equation (8) splits R into the squared norm of the change caused by the changed selection, $\|r\|^2$, and a cross term $2\langle e_{\text{frozen}}, r\rangle$. We measured both with D the relative squared distance of the last hidden state of the model, before and after the final normalization, in condition (iv) at 4 and 8 bits on the 64 WikiText-2 sequences of Section 3.3 (Table 13; the two terms sum to R within a relative error of 4×10^{-9}). At 8 bits, the magnitude of the cross term is at most 2.3% of R before the final normalization and at most 1.7% after it, so $R \approx \|r\|^2$. At 4 bits, the cross term reaches -39% of R (DeepSeek, before the normalization).

Table 13: The two terms of R (Eq. (8)) for the last hidden state (condition (iv)). Cross$/R$: cross term relative to R.

			Before normalization			After normalization	
Model	dtype	Bits	R	$\|r\|^2$	Cross$/R$	R	Cross$/R$
OLMoE	fp32	4	2.420×10^{-2}	2.841×10^{-2}	-17.4%	3.448×10^{-2}	-19.8%
		8	2.139×10^{-3}	2.187×10^{-3}	-2.26%	3.361×10^{-3}	-1.10%
DeepSeek	fp32	4	1.302×10^{-2}	1.813×10^{-2}	-39.2%	1.394×10^{-2}	-24.2%
		8	1.387×10^{-3}	1.376×10^{-3}	$+0.79\%$	1.567×10^{-3}	$+0.88\%$
Qwen3	fp16	4	2.312×10^{-1}	2.306×10^{-1}	$+0.26\%$	9.904×10^{-2}	-16.9%
		8	2.260×10^{-3}	2.286×10^{-3}	-1.15%	2.705×10^{-3}	-1.65%

E.4 Negative R

Because the cross term can be negative, so can R. In the layer-local measurement, $R \leq 0$ in 1 of the 1,910 combinations of layer and bit width (191 layers $\times$ 10 bit widths): one layer of Qwen1.5-MoE at 3 bits. At the model output, D is the KL divergence and Eq. (8) does not apply, but R is again the difference between two runs; it is negative in 3 of the 49 settings of condition (iv) (7 models $\times$ 7 bit widths), all at 3 bits or below: GPT-OSS at 3 bits ($m = -0.100$), Moonlight at 3 bits (-0.003), and OLMoE at 2 bits (-0.001). A negative R means that the free run is closer to the original model than the frozen run; Lee et al. (2026) also observe cases in which injecting the original routing increases the output error. We report these values as measured and use them only as differences between the two runs.

E.5 Comparison with prior experiments that restore the expert selection

Several prior works run a quantized model with the expert selection of the original model, or the reverse (Table 14). Our frozen run differs from them in three respects together. It fixes only the indices and leaves the gate values of the quantized router, so the two runs differ in the expert selection alone. It measures the deviation from the original model rather than the change in a capability metric (Section 2.3). It also spans 2 to 12 bits, which exposes the different powers of the two deviations (Section 4.2). The prior experiments fix the gate values as well or do not state how they are handled, judge the effect mostly with capability metrics, and measure a few settings at 4 bits or below.

The two directions of replay. For OLMoE, we also ran the reverse direction of Sakthivel et al. (2026): the original weights with the indices that the quantized model selects. For the output KL on

Table 14: Prior experiments that restore the original expert selection.

Work	Intervention	Gate values	Reference and metric	Bit widths
EAC-MoE (Chen et al., 2025)	Quantized model with the original model's selection scores, and the reverse	From the original scores	Likelihood of the reference text	3 bits
Gu (2026)	Four runs, including the quantized model with the clean route	From the clean run	NLL of the correct token	4-bit KV cache; 3- and 4-bit expert weights
RouteQuant (Fang & Huang, 2026)	Selected experts follow the full-precision router	Not stated	Accuracy	W4A4
SRA-MoE (Lee et al., 2026)	Original routing injected for a subset of tokens	Not stated	Divergence of the output distribution	3- and 4-bit weights
Sakthivel et al. (2026)	Original weights with the quantized model's routes (reverse)	–	NLL of the correct answer	4 bits
Ours	Quantized model with the original indices	Quantized router	Deviation from the original model	2–12 bits

WikiText-2, the reverse replay produces 0.868 [0.840, 0.904] of the deviation of the fully quantized model at 8 bits, and our frozen run removes 0.850 of it; for the option-distribution KL on MMLU-Pro, the two values are 0.877 [0.861, 0.892] and 0.90. For the continuous metrics, the two directions give nearly the same share. For answer agreement, they do not: the reverse replay produces 0.850 [0.762, 0.946] of the answer disagreement at 8 bits, whereas the frozen run removes 0.47 of it.

F The Constant κ and Aggregation across Layers

F.1 κ varies little across bit widths within a layer

Equation (5) writes the deviation of the frozen run in layer ℓ as $\kappa_\ell O_\ell$, where O_ℓ is the reconstruction error of the router weights on the router inputs of our measurement data. For each layer, we compute $\kappa_\ell = D_{\text{frozen},\ell}/O_\ell$ at every bit width of condition (i) and its coefficient of variation across bit widths. Table 15 reports the median of this coefficient over the layers of each model. Within the high-bit regime used for the law (Appendix I.1), it is 0.034–0.092; across all bit widths from 2 to 12, which include the low bit widths outside that regime, it is 0.080–0.339. Across the layers of a model, κ_ℓ varies more (coefficient of variation 0.288–0.567), and its layer median spans a factor of 122 across models (0.171 for GPT-OSS to 20.9 for Qwen3), so κ is a property of each layer, not a constant shared by layers or models. With GPTQ on the group-128 grid, the within-layer coefficient of variation in the high-bit regime is 0.009–0.039.

Table 15: Stability of $\kappa_\ell = D_{\text{frozen},\ell}/O_\ell$ (condition (i)). CV: coefficient of variation. Across bits: CV over the bit widths of a layer, median over layers. Group 128: in-regime CV on the group-128 grid.

			CV across bits		CV across	Group 128	
Model	Regime	κ (median)	In regime	2–12 bits	layers	RTN	GPTQ
OLMoE	6–12	1.672	0.040	0.171	0.567	0.037	0.039
Qwen1.5	6–12	7.889	0.047	0.261	0.329	0.047	0.016
DeepSeek	6–12	0.644	0.040	0.080	0.430	0.044	0.014
Qwen3	7–12	20.93	0.038	0.339	0.288	0.035	0.012
Moonlight	5–12	0.540	0.040	0.212	0.312	0.034	0.009
GPT-OSS	5–12	0.171	0.092	0.219	0.450	0.108	0.019
ERNIE	5–12	0.229	0.034	0.094	0.484	0.034	0.013

ERNIE stores its router weights in F32, a difference to note when the size of κ is compared across models. The coefficient of variation that Eq. (5) relies on compares κ_ℓ only across the bit widths of one layer.

F.2 Aggregation across layers

We aggregate every per-layer quantity by the median across layers (Section 3.2). Table 16 repeats the comparisons of Section 4 with the mean across layers. For the change rate f itself, the mean differs from the median by -5.0% to $+9.6\%$ in the layer-local condition (70 combinations of model and bit width) and by -14.7% to $+9.5\%$ at the model output (96 combinations), with the largest difference for DeepSeek at 8 bits at the model output (mean 0.119, median 0.140). The comparisons do not change: the ratio of h between models at 8 bits is 6.99 with the median and 6.76 with the mean, and the 3→2-bit ratios of f change by at most 0.082.

Table 16: Comparisons of Section 4 with the median (used in the paper) and the mean across layers (condition (i)).

Comparison	Median	Mean
h at 8 bits, largest / smallest model	6.99	6.76
f ratio 3→2 bits, GPT-OSS (the most sensitive)	1.444	1.362
f ratio 3→2 bits, other six models	1.002–1.625	1.006–1.614
Per-layer change in f, 8→10 bits	-74.4% to -75.2%	-74.2% to -75.1%

Why the median. A few layers lie far from the rest. At 8 bits, the ratio of the upper to the lower quartile of h across the layers of a model is 1.51–2.17, whereas the ratio of the largest to the smallest layer is 5.5–55.1. The last MoE layer has an h below the median of its model in all seven models (0.03–0.6 times the median), and in four models the leading layers are 10 to 50 times below it. The median keeps such layers from setting the value of a model; the mean gives the same comparisons (Table 16).

F.3 Bootstrap intervals

All intervals in this paper are 95% percentile intervals from 1,000 bootstrap resamples with seed 0, unless stated otherwise. On WikiText-2, the resampling unit is the sequence (64 sequences, resampled with replacement); on MMLU-Pro, it is the question (2,000 questions). In each resample, we sum the per-sequence contributions to recompute every per-layer quantity (D_{free}, D_{frozen}, R, f, h, m, and the fitted exponents), and then take the median across layers. When two conditions of the same model are compared (for example, GPTQ and RTN, or a quantized and a restored checkpoint), both use the same resampled indices, so the interval is that of a paired difference. Ratios between models resample each model independently. We compute a ratio only when the interval of its denominator excludes zero. The detection rates of Appendix H.3 use paired t intervals on subsamples instead (details there), and the proportions of Appendix D use Wilson score intervals.

G Models and Experimental Settings

G.1 Models

Table 17 lists the seven models. We chose publicly released MoE models that fit on a single node and that differ in how their routers score and select experts. Together they have 191 MoE layers. GPT-OSS-20B is released with its expert weights in MXFP4; we dequantize them to bf16 and use the result as the original model. Qwen3-30B-A3B also serves as the original model of the released checkpoints in Appendix C, and the held-out check of Appendix I.3 uses the Chat version of Qwen1.5-MoE in addition to the Base version.

The Hugging Face repositories are `Qwen/Qwen1.5-MoE-A2.7B`, `allenai/OLMoE-1B-7B-0924`, `deepseek-ai/DeepSeek-V2-Lite-Chat`, `Qwen/Qwen3-30B-A3B`, `moonshotai/Moonlight-16B-A3B`, `openai/gpt-oss-20b`, and `baidu/ERNIE-4.5-21B-A3B-Base-PT`.

Table 17: Models. Experts: routed experts per MoE layer. Score: function whose top-k entries select the experts. Shared: shared experts (kept in full precision).

Model	Version	Experts	k	Layers	Score	Router bias	Shared
Qwen1.5-MoE-A2.7B	Base	60	4	24	softmax	none	yes
OLMoE-1B-7B	Base	64	8	16	softmax	none	no
DeepSeek-V2-Lite	Chat	64	6	26	softmax	none	yes
Qwen3-30B-A3B	post-trained	128	8	48	softmax	none	no
Moonlight-16B-A3B	Base	64	6	26	sigmoid	selection only†	yes
GPT-OSS-20B	post-trained	32	4	24	logits*	on logits	no
ERNIE-4.5-21B-A3B	Base	64	6	27	softmax	selection only†	yes

*Gate values: softmax over the selected logits. †The bias shifts selection scores, not gate values.

G.2 Quantization

What is quantized. For the router, we quantize only the weight matrix W_g of each MoE layer; router biases stay in full precision. For the experts, we quantize the weights of the routed experts; shared experts, attention layers, embeddings, and the output layer stay in full precision (Appendix K.1 adds attention as a control).

Round-to-nearest. All bit-width sweeps use symmetric per-channel RTN, which needs no calibration data. For each output channel (row) w of a weight matrix, the scale is $s = \max_j |w_j|/(2^{b-1}-1)$ and each weight becomes $s \cdot \mathrm{round}(w_j/s)$. In the layer-local condition, we quantize on the GPU in fp32; in the model-output conditions, on the CPU in fp32. The relative weight error $\varepsilon = \|\hat{W} - W\|_F / \|W\|_F$ of the router weights halves with every additional bit: its slope in $\log_2$ per bit, fitted in each layer over $\{6, 7, 8, 9, 10, 12\}$ bits, has a median across layers of -1.0059 to -1.0071 in the seven models (Appendix I.2 explains the value).

GPTQ. The method comparison of Section 4.2 quantizes the router weights of each layer (condition (i)) with GPTQ (Frantar et al., 2023) and with RTN on the same grid: asymmetric, group size 128, no act-order. The only difference between the two is how each weight is rounded (Section 2.2). GPTQ uses the reference implementation of its authors. Its calibration inputs are the inputs of each router on 128 sequences of 2,048 tokens drawn at random from the WikiText-2 training split (seed 1234), accumulated as $H = 2\sum_t u_t u_t^\top / N$ in fp64. For dampening, 1% of the mean diagonal entry of H is added to every diagonal entry. For GPT-OSS, whose router input has 2,880 dimensions, the last group has 64 columns.

G.3 Precision and hardware

The layer-local measurement loads the weights in bf16 and computes the quantization and the recomputed layers in fp32. The model-output measurement holds the weights of both the original and the quantized model in fp32 and computes in fp32, with two exceptions. Qwen3-30B-A3B does not fit on our hardware in fp32, so it runs in fp16, both as the original model and as the quantized models; rounding the quantized weights to fp16 adds an error of 2.4% of the RTN error at 8 bits and 9.6% at 10 bits. GPT-OSS keeps its expert weights in bf16 and computes its router in fp32. The released checkpoints of Appendix C are dequantized and run in fp16 on WikiText-2 and in bf16 for generation. We ran all experiments on two NVIDIA RTX A6000 GPUs (48 GB each).

G.4 Data

WikiText-2. We split the WikiText-2 test split into sequences of 2,048 tokens with the tokenizer of each model and draw 64 of them with a random permutation (seed 1234). The deviations, the change rate, and the top-1 agreement use every position of a sequence except the initial one (2,047 per sequence, 131,008 in total); the NLL of the correct token uses the 2,046 positions that have a next token (130,944 positions).

MMLU-Pro. We draw 2,000 of the 12,032 questions of the MMLU-Pro test split, stratified by its 14 subject categories in proportion to their sizes (seed 0; each category is rounded and the total adjusted to 2,000). Of these, 1,647 questions have 10 options and 353 have 3 to 9. Sections 4.1 and 4.4 use the same 2,000 questions. For the option-letter scoring of Section 4.1, OLMoE, which has no chat template, receives the question and options as plain text followed by "Answer:"; DeepSeek and Qwen3 receive them in their chat templates, with thinking disabled for Qwen3. Each option letter has two single-token forms (with and without a leading space); the score of a letter is the log-sum-exp of the logits of both forms, and the model's answer is the letter with the highest score among the options of the question.

Bit widths. The layer-local sweep, condition (i), uses 2, 3, 4, 5, 6, 7, 8, 9, 10, and 12 bits in all seven models. Condition (iv) uses 2, 3, 4, 5, 6, 8, and 10 bits in all seven models. Condition (ii) uses 3 to 10 and 12 bits for Qwen1.5, OLMoE, and Qwen3; 3, 4, 5, 6, and 8 bits for DeepSeek; and 4, 5, 6, 8, and 10 bits for Moonlight, GPT-OSS, and ERNIE. Condition (iii) uses 4, 5, 6, and 8 bits in all seven models (Appendix K.3 adds 3 and 10 bits).

H Capability and Distance Metrics

H.1 Procedure

Models and comparisons. Section 4.1 quantizes OLMoE, DeepSeek-V2-Lite, and Qwen3-30B-A3B in condition (iv) (routed experts and routers) with RTN at 4 and 8 bits, and compares three runs: the original model, the free run, and the frozen run. "Quantization as a whole" is the change from the original model to the free run (free − original); "restoring the expert selection" is the change from the free run to the frozen run (frozen − free). Models run in fp32, except Qwen3 in fp16 (Appendix G.3).

MMLU-Pro. We give each of the 2,000 questions to the model once, zero-shot, with its options, and read the model's answer from the option-letter scores at the answer position (Appendix G.4); the option distribution is the softmax of these scores over the options of the question. Accuracy is the fraction of questions whose answer is the correct option. Answer disagreement is the fraction of questions whose answer differs from that of the original model; it counts changes from one incorrect answer to another. The NLL of the correct option is $-\log$ of its probability in the option distribution, and the option KL is $\mathrm{KL}(p \,\|\, \hat{p})$ between the option distributions of the original model, p, and of the quantized model, $\hat{p}$, as in Eq. (4).

WikiText-2. On the 131,008 positions of Appendix G.4, the output KL is the mean KL divergence between the next-token distributions of the original and the quantized model (Eq. (4)), and the top-1 disagreement is the fraction of positions whose most probable token differs from that of the original model. The NLL of the correct token is averaged over the 130,944 positions that have a next token.

Detection and cancellation. A metric detects a change if the 95% interval of the mean per-question (MMLU-Pro) or per-sequence (WikiText-2) difference excludes zero (paired bootstrap, Appendix F.3). For the NLL, we also report how much of the total change survives in the net change: with Δ_i the change for token or question i, the net change is $|\sum_i \Delta_i|$ and the total change is $\sum_i |\Delta_i|$.

H.2 Full results at 4 and 8 bits

Tables 18–21 give every comparison of Section 4.1. In all four, Q is quantization as a whole (free − original) and S is restoring the expert selection (frozen − free); brackets are 95% intervals, and pp means percentage points. Table 22 converts the distance metrics into the blind fraction m.

At 8 bits, the net change in accuracy is indistinguishable from zero in all six comparisons, although quantization as a whole changes 7.0–7.7% of the answers and restoring the expert selection changes 3.3–5.2 percentage points of the answer disagreement. The correct-to-incorrect and incorrect-to-correct changes nearly cancel (Qwen3: 30 and 30), and most changed answers move from one incorrect option to another (94 of Qwen3's 154), which accuracy does not count. Accuracy detects

Table 18: MMLU-Pro: accuracy and answers (2,000 questions). Changed answers between the compared runs: C→I, I→C, I→I (correct/incorrect). Original accuracy: OLMoE 0.1515, DeepSeek 0.1755, Qwen3 0.4580.

Model	Bits		ΔAccuracy (pp)	C→I · I→C · I→I	ΔDisagreement (pp)
OLMoE	4	Q	−1.20 [−2.75, +0.45]	146 · 122 · 806	+53.70 [51.40, 55.95]
		S	+0.15 [−1.10, +1.30]	76 · 79 · 498	−2.30 [−4.15, −0.20]
	8	Q	−0.15 [−0.60, +0.25]	11 · 8 · 121	+7.00 [5.90, 8.15]
		S	−0.05 [−0.45, +0.40]	10 · 9 · 94	−3.30 [−4.35, −2.30]
DeepSeek	4	Q	−0.25 [−1.75, +1.25]	110 · 105 · 526	+37.05 [35.05, 38.95]
		S	+0.25 [−0.90, +1.40]	67 · 72 · 346	−2.25 [−4.00, −0.50]
	8	Q	−0.50 [−1.10, +0.10]	25 · 15 · 103	+7.15 [6.05, 8.25]
		S	+0.35 [−0.25, +0.95]	15 · 22 · 95	−4.60 [−5.70, −3.45]
Qwen3	4	Q	−34.50 [−36.90, −31.70]	808 · 118 · 834	+88.00 [86.55, 89.40]
		S	+27.05 [24.50, 29.40]	123 · 664 · 893	−48.50 [−50.95, −45.90]
	8	Q	+0.00 [−0.70, +0.75]	30 · 30 · 94	+7.70 [6.55, 8.80]
		S	+0.40 [−0.30, +1.10]	24 · 32 · 82	−5.20 [−6.35, −4.15]

Table 19: MMLU-Pro: option distribution (same comparisons).

Model	Bits		ΔNLL of correct option	ΔOption KL
OLMoE	4	Q	+0.0587 [0.0358, 0.0813]	+0.0942 [0.0902, 0.0984]
		S	−0.0248 [−0.0387, −0.0101]	−0.0233 [−0.0259, −0.0208]
	8	Q	−0.0027 [−0.0067, +0.0009]	+0.0021 [0.0020, 0.0022]
		S	+0.0023 [−0.0011, +0.0060]	−0.0019 [−0.0020, −0.0018]
DeepSeek	4	Q	+0.0915 [0.0277, 0.1586]	+0.4075 [0.3842, 0.4306]
		S	−0.0328 [−0.0693, +0.0060]	−0.0808 [−0.0974, −0.0648]
	8	Q	−0.0392 [−0.0528, −0.0272]	+0.0139 [0.0128, 0.0150]
		S	+0.0167 [0.0051, 0.0294]	−0.0124 [−0.0134, −0.0114]
Qwen3	4	Q	−0.7414 [−1.1119, −0.4295]	+4.2173 [4.0993, 4.3336]
		S	+0.2125 [−0.0686, +0.5197]	−2.2716 [−2.4498, −2.0796]
	8	Q	+0.0749 [0.0171, 0.1384]	+0.1178 [0.0980, 0.1374]
		S	−0.0360 [−0.0975, +0.0218]	−0.1092 [−0.1287, −0.0896]

a change only in Qwen3 at 4 bits (in both comparisons), where the option KL of quantization as a whole is 4.22, about ten times the largest value among the other five settings (0.41).

H.3 Which metric detects the change

Table 23 marks, for each of the twelve comparisons, the metrics whose interval excludes zero on the full sample. The distance metrics detect all twelve. Accuracy detects two, both in Qwen3 at 4 bits. The NLL of the correct option detects seven and the NLL of the correct token on WikiText-2 detects seven, but when they detect a change, their sign does not tell whether the model moved away from the original: in Qwen3 at 4 bits, quantization lowers accuracy by 34.5 points while the NLL of the correct option improves by 0.74 (Appendix H.5).

Why answer agreement detects the change and accuracy does not. In the comparison that restores the expert selection, the change in agreement with the original model is, for each question, -1, 0, or $+1$, as is the change in correctness. Both are signed and can cancel across questions. Answer agreement still detects the change and accuracy does not, because they use different references: a question whose answer moves toward the original answer counts toward agreement whether or not the original answer was correct, whereas accuracy counts only moves to or from the correct option, and those moves cancel (Table 18).

Detection rate versus the number of questions. To see whether more questions would help accuracy, we drew 1,000 random subsamples of n questions (n = 50, 100, 200, 500, 1,000) or sequences

Table 20: WikiText-2: NLL of the correct token (64 sequences). Net/total: $|\sum_i \Delta_i| / \sum_i |\Delta_i|$ over tokens.

Model	Bits		ΔNLL of correct token	Net/total
OLMoE	4	Q	+0.1642 [0.1349, 0.2033]	0.440
		S	−0.0214 [−0.0277, −0.0164]	0.130
	8	Q	+0.00043 [0.00005, 0.00084]	0.011
		S	−0.00024 [−0.00063, +0.00016]	0.007
DeepSeek	4	Q	+0.0751 [0.0679, 0.0819]	0.169
		S	−0.0047 [−0.0073, −0.0022]	0.023
	8	Q	−0.00014 [−0.00091, +0.00066]	0.002
		S	−0.00011 [−0.00086, +0.00060]	0.002
Qwen3	4	Q	+0.5169 [0.4497, 0.6010]	0.569
		S	−0.4155 [−0.4960, −0.3479]	0.537
	8	Q	−0.00055 [−0.00172, +0.00074]	0.008
		S	+0.00099 [−0.00020, +0.00209]	0.016

Table 21: WikiText-2: distance metrics (same comparisons).

Model	Bits		ΔOutput KL	ΔTop-1 disagreement (pp)
OLMoE	4	Q	+0.1696 [0.1427, 0.2059]	+18.20 [17.44, 19.06]
		S	−0.0334 [−0.0406, −0.0281]	−1.86 [−2.04, −1.68]
	8	Q	+0.00309 [0.00258, 0.00374]	+2.28 [2.17, 2.39]
		S	−0.00263 [−0.00319, −0.00219]	−1.28 [−1.36, −1.20]
DeepSeek	4	Q	+0.2191 [0.2093, 0.2291]	+18.30 [17.84, 18.76]
		S	−0.0414 [−0.0437, −0.0393]	−1.95 [−2.11, −1.82]
	8	Q	+0.00430 [0.00405, 0.00458]	+2.53 [2.41, 2.64]
		S	−0.00372 [−0.00397, −0.00350]	−1.54 [−1.63, −1.45]
Qwen3	4	Q	+0.5812 [0.5053, 0.6735]	+27.54 [26.04, 29.22]
		S	−0.4362 [−0.5282, −0.3616]	−12.75 [−14.42, −11.22]
	8	Q	+0.00537 [0.00488, 0.00600]	+2.67 [2.57, 2.77]
		S	−0.00475 [−0.00538, −0.00427]	−1.66 [−1.75, −1.56]

(n = 4, 8, 16, 32) without replacement and counted how often a paired t interval (95%) of the mean difference excludes zero (Figure 5; at the full sample, there is a single test). For restoring the expert selection at 8 bits, the option KL detects the change in all 1,000 subsamples of 200 questions in all three models, whereas accuracy does not detect it even with all 2,000 questions. On WikiText-2, the output KL detects the same change in all subsamples of 16 sequences, whereas the NLL of the correct token does not detect it with all 64. Running the original model twice gives a detection rate of zero for every metric and size.

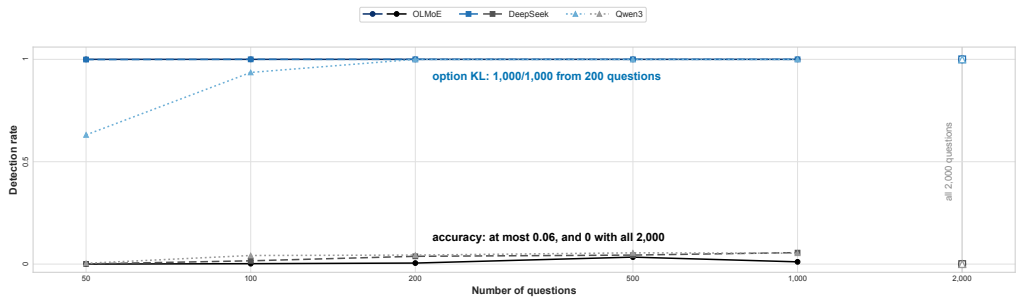

Figure 5: Detection rate of restoring the expert selection at 8 bits, versus the number of MMLU-Pro questions (1,000 random subsamples per point; one test at 2,000, hollow markers).

Table 22: Blind fraction m with each distance metric: |S| divided by Q (point estimates above).

	4 bits			8 bits		
Distance metric	OLMoE	DeepSeek	Qwen3	OLMoE	DeepSeek	Qwen3
Answer disagreement (MMLU-Pro)	0.04	0.06	0.55	0.47	0.64	0.68
Option KL (MMLU-Pro)	0.25	0.20	0.54	0.90	0.89	0.93
Output KL (WikiText-2)	0.20	0.19	0.75	0.85	0.87	0.89
Top-1 disagreement (WikiText-2)	0.10	0.11	0.46	0.56	0.61	0.62

Table 23: Which metric detects each change (full sample; ✓: 95% interval excludes zero).

			MMLU-Pro				WikiText-2		
Model	Bits		Accuracy	NLL	Disagreement	KL	NLL	KL	Top-1
OLMoE	4	Q	–	✓	✓	✓	✓	✓	✓
		S	–	✓	✓	✓	✓	✓	✓
	8	Q	–	–	✓	✓	✓	✓	✓
		S	–	–	✓	✓	–	✓	✓
DeepSeek	4	Q	–	✓	✓	✓	✓	✓	✓
		S	–	–	✓	✓	✓	✓	✓
	8	Q	–	✓	✓	✓	–	✓	✓
		S	–	✓	✓	✓	–	✓	✓
Qwen3	4	Q	✓	✓	✓	✓	✓	✓	✓
		S	✓	–	✓	✓	✓	✓	✓
	8	Q	–	✓	✓	✓	–	✓	✓
		S	–	–	✓	✓	–	✓	✓
Detected			2/12	7/12	12/12	12/12	7/12	12/12	12/12

H.4 OLMoE FROM 4 TO 10 BITS

Section 4.1 states that in OLMoE the six settings from 4 to 10 bits change 3.7–53.7% of the answers while the net change in accuracy stays indistinguishable from zero. Table 24 and Figure 6 give the values.

Table 24: OLMoE quantized as a whole, 4 to 10 bits (2,000 MMLU-Pro questions). ε: weight error of the routers and experts together. Brackets: 95% intervals.

Bits	ε	Answers changed	C→I · I→C	ΔAccuracy (pp)
4	0.1556	53.7% [51.4, 56.0]	146 · 122	-1.20 [-2.75, $+0.45$]
5	0.0727	25.5% [23.6, 27.5]	60 · 58	-0.10 [-1.25, $+0.90$]
6	0.0352	16.3% [14.7, 18.0]	33 · 42	$+0.45$ [-0.40, $+1.30$]
7	0.0173	10.8% [9.5, 12.1]	20 · 21	$+0.05$ [-0.55, $+0.65$]
8	0.0086	7.0% [5.9, 8.2]	11 · 8	-0.15 [-0.60, $+0.25$]
10	0.0021	3.7% [2.9, 4.5]	7 · 5	-0.10 [-0.45, $+0.25$]

OLMoE answers 15.2% of the questions correctly (chance is about 10.6% with 9.44 options on average), so most of its answers are incorrect before quantization. Qwen3 (45.8% correct) shows the same pattern at 8 bits (Table 18), so the cancellation is not limited to a model with low accuracy.

H.5 QWEN3 AT 4 BITS: ACCURACY AND THE NLL DISAGREE IN SIGN

In Qwen3 at 4 bits, quantization lowers accuracy from 0.458 to 0.113 but lowers the mean NLL of the correct option by 0.74. Grouping the questions by how their answer changes decomposes this net change. The 808 questions that change from correct to incorrect raise the mean NLL by 2.26, and the 108 that stay correct raise it by 0.02. The 1,084 questions that the original model answers incorrectly lower it by 3.02: 834 that move to another incorrect option (-1.88), 132 that keep the same incorrect option (-0.47), and 118 that become correct (-0.67). The NLL of the correct option thus reads a model that disagrees with the original on 88% of the questions as better than the original. Over the

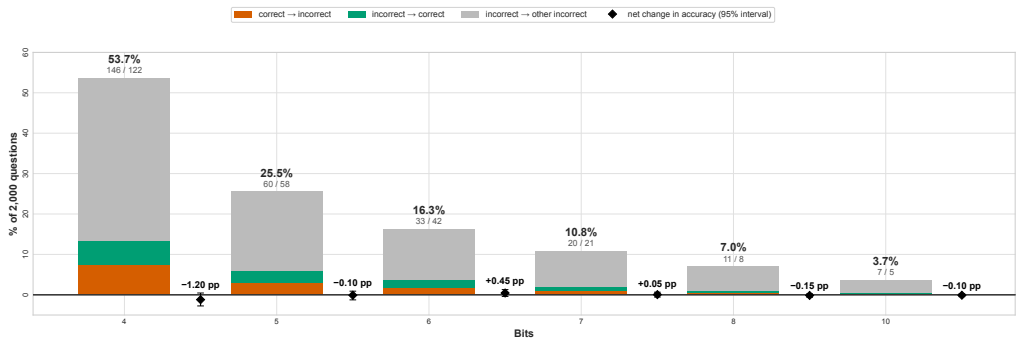

Figure 6: OLMoE quantized as a whole, 4 to 10 bits (2,000 MMLU-Pro questions). Bars: changed answers by type, labeled with their share and the counts of correct-to-incorrect / incorrect-to-correct changes. Diamonds: net change in accuracy with 95% intervals. The change in accuracy is indistinguishable from zero at every bit width.

same questions, the mean entropy of the option distribution rises from 0.24 to 0.73 nats. Restoring the expert selection raises accuracy to 0.3835; the remaining gap of 7.5 points to the original model persists with the original selection, so it comes from the quantized experts and gate values.

I The High-Bit Law

I.1 The high-bit regime

The derivation below (I.2) keeps only the leading terms in the weight error ε. This approximation holds when few tokens change their selection and each change swaps the k-th and $(k{+}1)$-th experts; at 8 bits, 88–99% of the experts that leave a selection were ranked k-th (Appendix J.2). We call a bit width *high* when its change rate is at most 0.30 (condition (i), median across layers), and we fit Eq. (6) only there: 7–12 bits for Qwen3; 6–12 bits for Qwen1.5-MoE, OLMoE, and DeepSeek; and 5–12 bits for Moonlight, GPT-OSS, and ERNIE. The exponents use $\{7, 8, 9, 10, 12\}$ bits, inside the regime for all seven models. The regime is not the set of bit widths above the crossover b^*: Moonlight and GPT-OSS have b^* of 1.83 and 1.33 (Table 26), but their change rates exceed 0.30 below 5 bits.

Table 25 shows why the fit stops there. Inside the regime, the largest residual is 0.004–0.016. Just below it, the residuals stay within 0.02, and every residual above 0.05 lies at 2 to 4 bits, where $f \geq 0.70$. At 2 bits, m lies 0.11–0.35 above the curve in all seven models and exceeds its 3-bit value in five; this is the step in which R grows 2.0–14.1-fold while f grows by a factor of at most 1.6 (Section 4.3). The threshold of 0.30 is thus conservative, and we use one rule for all models.

Table 25: Residuals of Eq. (6) (measured m minus the curve), condition (i). In regime: largest magnitude.

			Outside the regime				
Model	b^*	In regime	2 bits	3 bits	4 bits	5 bits	6 bits
OLMoE	4.32	0.016	+0.184	+0.106	+0.075	+0.005	
Qwen1.5	5.23	0.013	+0.331	+0.158	+0.033	−0.002	
DeepSeek	4.72	0.015	+0.107	+0.046	−0.009	−0.019	
Qwen3	4.77	0.004	+0.352	+0.102	+0.111	+0.029	−0.010
Moonlight	1.83	0.007	+0.213	−0.020	−0.015		
GPT-OSS	1.33	0.007	+0.234	−0.010	−0.018		
ERNIE	4.36	0.008	+0.325	+0.026	+0.008		

I.2 WHY THE TWO DEVIATIONS SCALE WITH DIFFERENT POWERS OF ε

Take one layer in condition (i). Quantizing its router shifts the scores of token t by $\delta z_t = (\hat{W}_g - W_g)\, u_t$, which is proportional to ε. We assume that (A1) the direction of the weight error is stable across bit widths, (A2) the gap Δ_t between the k-th and $(k{+}1)$-th scores has a finite density at zero, (A3) the two experts at this boundary produce different outputs, and (A4) the gate values are smooth in the scores.

f is linear in ε. For small ε, a token's selection changes only when the shift closes its gap: $\Delta_t < \varepsilon a_t$, where a_t is the rate at which the k-th and $(k{+}1)$-th scores approach each other. By (A2), the fraction of such tokens is $f = \lambda\varepsilon + o(\varepsilon)$. From 8 to 10 bits, where ε falls to about a quarter, the median change rate across layers falls by 74.3–75.5% in the seven models.

R is linear in ε. A changed token swaps its k-th expert for its $(k{+}1)$-th. At the swap, their scores and gate values are equal, so the output jumps by $J_t = g_t\,[E_{(k+1)}(u_t) - E_{(k)}(u_t)]$, which does not shrink with ε (A3). In Eq. (8), $r_t = J_t$ for changed tokens and $r_t = 0$ otherwise, so $\|r\|^2$ is f times the mean squared jump. The cross term is $O(\varepsilon^2)$, since $e_{\text{frozen}} = O(\varepsilon)$ and r is nonzero on only an $O(\varepsilon)$ fraction of tokens. Hence

$$R = A\varepsilon + o(\varepsilon), \qquad h = R/f \to A/\lambda, \tag{9}$$

so the deviation per selection change tends to a constant of the layer: the mean squared jump at the boundary. In Moonlight and ERNIE, the router bias enters the selection scores but not the gate values (Table 17), so the two gate values differ at the swap; the jump is still $O(1)$, and the scaling is the same.

D_{frozen} is quadratic in ε. With the selection fixed, the output is smooth in the scores (A4), so its change is linear in δz_t and $D_{\text{frozen}} = B'\varepsilon^2 + o(\varepsilon^2)$. Because O_ℓ is also quadratic in ε, $\kappa_\ell = D_{\text{frozen},\ell}/O_\ell$ does not depend on ε under (A1).

Hence $\alpha_{\text{frozen}} = 2$ and $\alpha_R = 1$ to leading order; the fits give 1.970–2.002 and 0.982–1.013 (Table 26). With the selection fixed, the squared deviation is a smooth function of the weights whose expansion starts at second order, like the reconstruction error of Eq. (2). R instead comes from an $O(\varepsilon)$ fraction of tokens whose output jumps by $O(1)$.

From the two powers to Eq. (6). Dividing the two, $m/(1-m) = R/D_{\text{frozen}} \approx (A/B')\,\varepsilon^{-1}$. If $\varepsilon \propto 2^{-sb}$, then $\log_2[m/(1-m)] = s(\alpha_{\text{frozen}} - \alpha_R)(b - b^*)$. The measured slope of $\log_2[m/(1-m)]$ per bit is 0.992–1.013 (median across layers over $\{7, 8, 9, 10, 12\}$ bits), which gives Eq. (6).

Why ε halves with every bit. Symmetric RTN with b bits has a step of $\max_j |w_j|/(2^{b-1}-1)$. If the rounding error is uniform within a step, $\varepsilon \propto 1/(2^{b-1}-1)$, whose $\log_2$ has a slope of -1.0069 per bit over $\{6, 7, 8, 9, 10, 12\}$ bits; the measured slopes are -1.0059 to -1.0071. At 3 to 5 bits, where ε reaches 0.75, the slope is -0.60 to -1.16.

Checking the assumptions. (A2): the density of the gap at zero, scaled by the median gap, is 0.68–0.78, and with one constant per model, the fraction of tokens whose gap is below that constant times ε matches the measured change rate within 5.5% in the high-bit regime (six models; Qwen1.5-MoE not measured). (A1) agrees with the stable κ_ℓ (Appendix F.1), and (A3) with the constant h at high bit widths (Appendix J.4).

I.3 EXPONENTS AND THE LAW IN EACH MODEL

Table 26 gives the exponents, crossover bit widths, and blind fractions of each model with their intervals. For each of the 191 layers, we fit both exponents on $\{7, 8, 9, 10, 12\}$ bits; the Spearman correlation of the per-layer exponents with layer depth is -0.28 to $+0.10$ for α_{frozen} and -0.16 to $+0.22$ for α_R, so the exponents do not depend on depth. Fitted instead over the whole high-bit regime of each model, the exponents are 1.990–2.005 and 0.988–1.004.

Table 26: High-bit law. α: exponents in condition (i) over $\{7, 8, 9, 10, 12\}$ bits. b^*: crossover bit width. m: blind fraction at 8 bits (model output: condition (iv)). Δm: GPTQ − RTN at 7 bits (group size 128). Super- and subscripts: 95% interval. †Beyond the measured bit widths. ‡Not fp32.

Model	α_{frozen}	α_R	b^*	m (layer-local)	m (model output)	Δm (GPTQ)
OLMoE	$1.990^{+0.006}_{-0.002}$	$0.998^{+0.011}_{-0.008}$	$4.32^{+0.07}_{-0.08}$	$0.921^{+0.005}_{-0.004}$	$0.850^{+0.011}_{-0.017}$	$+0.009^{+0.005}_{-0.003}$
Qwen1.5	$2.000^{+0.002}_{-0.005}$	$0.999^{+0.012}_{-0.011}$	$5.23^{+0.04}_{-0.03}$	$0.875^{+0.004}_{-0.004}$	$0.783^{+0.005}_{-0.004}$	$+0.026^{+0.005}_{-0.009}$
DeepSeek	$1.994^{+0.005}_{-0.001}$	$1.001^{+0.011}_{-0.011}$	$4.72^{+0.04}_{-0.03}$	$0.906^{+0.005}_{-0.004}$	$0.866^{+0.006}_{-0.006}$	$+0.010^{+0.002}_{-0.003}$
Qwen3	$1.999^{+0.002}_{-0.004}$	$0.988^{+0.006}_{-0.004}$	$4.77^{+0.04}_{-0.06}$	$0.901^{+0.007}_{-0.002}$	$0.886^{+0.011}_{-0.011}$ ‡	$+0.010^{+0.001}_{-0.002}$
Moonlight	$2.002^{+0.004}_{-0.001}$	$1.013^{+0.019}_{-0.018}$	$1.83^{+0.05}_{-0.10}$ †	$0.987^{+0.000}_{-0.001}$	$0.806^{+0.009}_{-0.010}$	$+0.004^{+0.001}_{-0.001}$
GPT-OSS	$1.970^{+0.008}_{-0.002}$	$0.982^{+0.019}_{-0.015}$	$1.33^{+0.03}_{-0.04}$ †	$0.991^{+0.000}_{-0.001}$	$0.612^{+0.012}_{-0.012}$ ‡	$+0.002^{+0.001}_{-0.000}$
ERNIE	$1.999^{+0.003}_{-0.003}$	$0.998^{+0.013}_{-0.013}$	$4.36^{+0.11}_{-0.10}$	$0.925^{+0.005}_{-0.006}$	$0.718^{+0.025}_{-0.026}$	$+0.010^{+0.003}_{-0.002}$

The blind fraction at every bit width. Table 27 gives the layer-local blind fraction of each model at every bit width. Within the high-bit regime, m increases with every step in bit width in all seven models (37 of 37 adjacent pairs), and the fraction of the deviation that the reconstruction error captures, $1 - m$, falls by a factor of 3.28–4.19 from 8 to 10 bits. At 8 bits, $1 - m$ is 0.009–0.125, so the reconstruction error captures at most 12.5% of the deviation in the layer-local condition; at the model output, where every layer is quantized, it captures 11–39% (condition (iv), Table 26).

Table 27: Blind fraction m by bit width (condition (i), median across layers). Super- and subscripts: 95% interval. Bold: high-bit regime.

Bits	OLMoE	Qwen1.5	DeepSeek	Qwen3	Moonlight	GPT-OSS	ERNIE
2	$.3510^{.3582}_{.3427}$	$.4277^{.4333}_{.4151}$	$.2388^{.2455}_{.2338}$	$.4792^{.4890}_{.4681}$	$.7436^{.7505}_{.7375}$	$.8486^{.8514}_{.8451}$	$.4882^{.4942}_{.4744}$
3	$.3928^{.4112}_{.3847}$	$.3338^{.3399}_{.3287}$	$.2794^{.2857}_{.2723}$	$.3286^{.3450}_{.3182}$	$.6726^{.6826}_{.6624}$	$.7520^{.7568}_{.7495}$	$.3059^{.3152}_{.2890}$
4	$.5198^{.5283}_{.5104}$	$.3317^{.3404}_{.3266}$	$.3694^{.3787}_{.3637}$	$.4801^{.4846}_{.4644}$	$.8033^{.8110}_{.7956}$	$.8462^{.8501}_{.8426}$	$.4462^{.4591}_{.4281}$
5	$.6211^{.6358}_{.6066}$	$.4587^{.4651}_{.4536}$	$.5296^{.5443}_{.5249}$	$.5681^{.5749}_{.5569}$	$\mathbf{.8929}^{.8989}_{.8889}$	$\mathbf{.9202}^{.9229}_{.9190}$	$\mathbf{.6012}^{.6155}_{.5828}$
6	$\mathbf{.7733}^{.7832}_{.7636}$	$\mathbf{.6172}^{.6238}_{.6123}$	$\mathbf{.6932}^{.6992}_{.6886}$	$.6906^{.6963}_{.6838}$	$\mathbf{.9468}^{.9496}_{.9440}$	$\mathbf{.9621}^{.9631}_{.9607}$	$\mathbf{.7534}^{.7660}_{.7374}$
7	$\mathbf{.8494}^{.8572}_{.8419}$	$\mathbf{.7794}^{.7863}_{.7703}$	$\mathbf{.8184}^{.8236}_{.8156}$	$\mathbf{.8200}^{.8252}_{.8151}$	$\mathbf{.9728}^{.9743}_{.9720}$	$\mathbf{.9803}^{.9811}_{.9799}$	$\mathbf{.8604}^{.8700}_{.8503}$
8	$\mathbf{.9213}^{.9257}_{.9167}$	$\mathbf{.8752}^{.8787}_{.8712}$	$\mathbf{.9061}^{.9111}_{.9020}$	$\mathbf{.9014}^{.9075}_{.8995}$	$\mathbf{.9866}^{.9875}_{.9859}$	$\mathbf{.9907}^{.9911}_{.9902}$	$\mathbf{.9250}^{.9298}_{.9191}$
9	$\mathbf{.9654}^{.9669}_{.9636}$	$\mathbf{.9334}^{.9353}_{.9312}$	$\mathbf{.9535}^{.9551}_{.9512}$	$\mathbf{.9494}^{.9511}_{.9476}$	$\mathbf{.9932}^{.9937}_{.9929}$	$\mathbf{.9952}^{.9955}_{.9951}$	$\mathbf{.9599}^{.9635}_{.9576}$
10	$\mathbf{.9809}^{.9825}_{.9792}$	$\mathbf{.9620}^{.9638}_{.9602}$	$\mathbf{.9765}^{.9773}_{.9752}$	$\mathbf{.9748}^{.9756}_{.9739}$	$\mathbf{.9967}^{.9970}_{.9965}$	$\mathbf{.9978}^{.9979}_{.9976}$	$\mathbf{.9812}^{.9824}_{.9793}$
12	$\mathbf{.9955}^{.9959}_{.9951}$	$\mathbf{.9913}^{.9918}_{.9904}$	$\mathbf{.9938}^{.9942}_{.9933}$	$\mathbf{.9935}^{.9939}_{.9932}$	$\mathbf{.9992}^{.9992}_{.9991}$	$\mathbf{.9994}^{.9994}_{.9993}$	$\mathbf{.9953}^{.9957}_{.9947}$

Held-out bit widths. We fitted b^* on some bit widths of the high-bit regime and compared Eq. (6) with m measured at the other bit widths (Table 28). At the eight held-out points, the curve differs from the measured m by at most 0.0146, that is, by 0.7–12.2% of $1 - m$ (median 6.8%). The relative error is largest at 12 bits only because $1 - m$ is below 0.01 there. This check used the Chat version of Qwen1.5-MoE, which our other measurements replaced with the Base model; repeating it on the Base model, with b^* fitted on 6, 8, and 10 bits, gives differences of 0.0122, 0.0039, and 0.0007 at 7, 9, and 12 bits.

Table 28: Eq. (6) at bit widths held out from the fit.

Model	Fit on	b^*	Held out	Curve	Measured	$\lvert\Delta m\rvert/(1-m)$
Qwen3	8, 10	4.77	7	0.824	0.8200	2.2%
			9	0.949	0.9494	0.7%
			12	0.993	0.9935	7.7%
OLMoE	6, 8, 10	4.33	7	0.864	0.8494	9.7%
			9	0.962	0.9654	9.7%
			12	0.995	0.9955	12.2%
Qwen1.5 (Chat)	6, 7, 8, 10	4.88	9	0.946	0.9471	2.0%
			12	0.993	0.9934	5.8%

I.4 GPTQ

Section 4.2 compares GPTQ with RTN on the same grid (group size 128, asymmetric). Table 29 gives the per-model values. GPTQ lowers the reconstruction error of the router weights, O, to 0.48–0.68 of that of RTN at 4 bits and 0.475–0.683 at 8 bits, while it raises the weight error ε by a factor of 1.10–1.16. The constant κ of Eq. (5) stays within 0.99–1.09 of its RTN value, so $D_{\text{frozen}} = \kappa O$ falls with O. Over $\{7, 8, 9, 10, 12\}$ bits, the exponents with GPTQ are $\alpha_{\text{frozen}} = 1.997$–$2.003$ and $\alpha_R = 0.992$–1.011, and those with RTN on the same grid are 1.998–2.009 and 0.994–1.005.

To compare the two deviations in one fit, we fit $D_{\text{free}} = A\varepsilon + B\varepsilon^2$ in each layer on $\{3, 4, 5, 6, 8, 10\}$ bits (weighted by $1/D_{\text{free}}^2$); by Appendix I.2, the linear term is the leading term of R and the quadratic term that of D_{frozen}. From RTN to GPTQ, B falls to 0.31–0.58 of its value but A only to 0.59–0.75, and the ratio for A is the square root of the ratio for B within 6% (0.947–1.054) in all seven models. The same relation holds along the bit axis: one more bit divides D_{frozen} by 4 and R by 2. GPTQ thus reduces the routing-induced deviation less than the deviation that the reconstruction error captures, and the blind fraction grows (Table 30): at 4 bits, Δm is $+0.017$ to $+0.074$ (all intervals exclude zero), and the crossover bit width of this two-term fit, which uses the group-128 grid and so differs from the b^* of Table 26, falls by 0.19–0.62 bits. For four models (OLMoE, Qwen3, Moonlight, GPT-OSS), the crossover lies below 3 bits for at least one method, outside the measured range, so the value is an extension of the fit.

Table 29: GPTQ relative to RTN (group size 128; ratios GPTQ/RTN, median across layers). A, B: coefficients of $D_{\text{free}} = A\varepsilon + B\varepsilon^2$. b^*: crossover of this fit, RTN $\to$ GPTQ. All intervals of the O ratios exclude 1.

Model	O (4 bits)	O (8)	ε (8)	κ (8)	B	A	$A/\sqrt{B}$	b^*	Δm (4 bits) [95%]
OLMoE	0.681	0.683	1.105	1.051	0.570	0.748	0.976	2.73 $\to$ 2.36	$+0.034$ [0.029, 0.039]
Qwen1.5	0.627	0.623	1.116	1.086	0.558	0.703	0.949	4.07 $\to$ 3.61	$+0.074$ [0.066, 0.079]
DeepSeek	0.640	0.632	1.099	1.084	0.579	0.718	0.947	3.52 $\to$ 3.30	$+0.057$ [0.039, 0.061]
Qwen3	0.624	0.622	1.125	1.058	0.463	0.719	1.036	3.20 $\to$ 2.81	$+0.038$ [0.033, 0.044]
Moonlight	0.514	0.524	1.108	1.061	0.411	0.635	0.997	1.44 $\to$ 1.10	$+0.023$ [0.020, 0.026]
GPT-OSS	0.484	0.475	1.157	0.990	0.307	0.586	1.054	0.57 $\to -0.05$	$+0.017$ [0.016, 0.018]
ERNIE	0.619	0.627	1.116	1.049	0.534	0.692	0.961	3.32 $\to$ 3.14	$+0.040$ [0.027, 0.045]

Table 30: $\Delta m = m_{\text{GPTQ}} - m_{\text{RTN}}$ by bit width (group size 128). Underlined: 95% interval includes zero.

Model	3	4	5	6	7	8	9	10	12
OLMoE	$\underline{+.033}$	$+.034$	$+.022$	$\underline{+.001}$	$+.009$	$\underline{+.002}$	$+.002$	$\underline{+.000}$	$\underline{+.000}$
Qwen1.5	$+.043$	$+.074$	$+.020$	$+.025$	$+.026$	$+.012$	$+.006$	$+.005$	$\underline{+.000}$
DeepSeek	$+.015$	$+.057$	$+.010$	$\underline{-.001}$	$+.010$	$+.006$	$\underline{+.000}$	$\underline{+.001}$	$\underline{+.000}$
Qwen3	$+.049$	$+.038$	$+.025$	$+.016$	$+.010$	$+.004$	$+.003$	$+.001$	$+.000$
Moonlight	$+.032$	$+.023$	$+.010$	$+.006$	$+.004$	$+.001$	$+.001$	$+.000$	$+.000$
GPT-OSS	$+.024$	$+.017$	$+.009$	$+.005$	$+.002$	$+.001$	$+.001$	$+.000$	$+.000$
ERNIE	$+.028$	$+.040$	$+.031$	$+.011$	$+.010$	$+.004$	$+.002$	$\underline{+.001}$	$+.000$

J The Change Rate and the Deviation per Selection Change

J.1 Decomposing the growth of R

In each layer, the routing-induced deviation factors into the change rate and the deviation per selection change, $R = f \cdot h$ (Eq. (7)). Counting changed experts instead of changed tokens gives a second factorization, $R = c \cdot g$, where c is the number of changed experts per token and $g = R/c$ is the deviation per changed expert. Together, $R = f \cdot (c/f) \cdot g$, where c/f is the number of changed experts per changed token. When the bit width decreases by one step, the increase in $\log R$ splits exactly into the increases in $\log f$, $\log(c/f)$, and $\log g$; we call each part's fraction of the increase in $\log R$ its share. The shares sum to one in every layer (up to 10^{-12}), and Table 31 reports their medians across layers, which need not sum to one.

Table 31: Shares of the increase in $\log R$ per step of the bit grid (condition (i); median across layers, and across steps for High, Middle, and Low). High: both bit widths ≥ 6; Middle: 6→5, 5→4; Low: 4→3, 3→2. Brackets: interquartile range.

	Share of f			Shares, 3→2 bits		
Model	High	Middle	Low	f	c/f	g
OLMoE	0.982 [0.950, 1.004]	0.732	0.153	0.056	0.623	0.320
Qwen1.5	0.977 [0.929, 1.005]	0.657	0.134	0.058	0.434	0.509
DeepSeek	0.974 [0.940, 1.006]	0.707	0.144	0.057	0.606	0.336
Qwen3	0.968 [0.927, 0.994]	0.520	0.023	0.003	0.561	0.424
Moonlight	0.987 [0.958, 1.016]	0.926	0.410	0.173	0.321	0.513
GPT-OSS	1.000 [0.967, 1.029]	0.921	0.364	0.219	0.381	0.347
ERNIE	0.985 [0.956, 1.017]	0.873	0.115	0.066	0.312	0.611

At high bit widths, the change rate carries 97–100% of the increase, so within a model f follows R. The share falls as the bit width decreases, and in the step from 3 to 2 bits it is 0.3–21.9% (at most 6.6% in five models); the rest splits between more changed experts per changed token (c/f, 31–62%) and a larger deviation per changed expert (g, 32–61%).

J.2 Which experts change

Table 32 ranks the experts that leave and enter the selection by the scores of the original model. At 8 bits, 88–99% of the experts that leave were ranked k-th and 86–97% of those that enter were ranked $(k{+}1)$-th, so almost every change exchanges the two experts at the boundary. At 2 bits, only 14–32% of the leaving experts were ranked k-th and 4.0–13.5% of the entering experts were ranked $(k{+}1)$-th; the entering experts have a median original rank of 9 to 28.5. A token whose selection changes loses, in original gate weight, 6.0–17.5% of its selection at 8 bits and 30–73% at 2 bits, 2.8–9.0 times as much.

Table 32: Original ranks of the experts that leave or enter the selection (condition (i)). Lost gate weight: original gate weight of the leaving experts, per changed token. Brackets: 95% intervals.

		Leaving: ranked k-th		Entering: ranked $(k{+}1)$-th		Entering:	Lost gate weight
Model	k	8 bits	2 bits	8 bits	2 bits	median rank, 2 bits	2 bits / 8 bits
Qwen1.5	4	0.957	0.269	0.909	0.049	21.5	5.08 [5.05, 5.13]
OLMoE	8	0.945	0.170	0.924	0.058	21.5	6.66 [6.59, 6.73]
DeepSeek	6	0.950	0.198	0.929	0.064	17.5	6.20 [6.14, 6.26]
Qwen3	8	0.883	0.142	0.856	0.040	28.5	9.04 [8.88, 9.15]
Moonlight	6	0.986	0.284	0.971	0.072	19.0	4.87 [4.79, 4.94]
GPT-OSS	4	0.983	0.323	0.970	0.135	9.0	2.79 [2.76, 2.81]
ERNIE	6	0.972	0.230	0.951	0.068	19.0	6.73 [6.64, 6.84]

J.3 The step from 3 to 2 bits

Table 33 gives the factors behind Section 4.3. From 3 to 2 bits, the change rate grows by at most a factor of 1.63, the number of changed experts per token by 1.5–3.8, and the routing-induced deviation by 2.0–14.1. In 183 of the 191 layers [181, 185], R grows faster than c, so counting changed experts does not follow the deviation either; the deviation per changed expert, g, grows by 1.31–4.72.

The expert selection is not random at 2 bits. Even where the change rate saturates, the selection keeps much of the original. At 2 bits, 45–76% of the k selected experts change per token (median across layers), so 25–55% of the original experts stay selected. If the quantized router selected k of its E experts at random, only k/E of them, 6.25–12.5%, would stay. The Jaccard distance between the original and quantized selections is 0.60–0.84 at 2 bits, below its maximum of 1. Figure 7 shows the fraction of changed experts at every bit width.

Table 33: Growth from 3 to 2 bits (condition (i); 95% intervals). The h column is the ratio of the medians across layers (no interval). Last column: layers in which R grows faster than c.

Model	f	c	R	g	h	R faster than c
Qwen3	1.002 [1.002, 1.002]	1.54 [1.52, 1.56]	2.01 [1.90, 2.11]	1.31 [1.25, 1.39]	1.93	42/48
DeepSeek	1.061 [1.058, 1.062]	2.00 [1.99, 2.01]	2.60 [2.57, 2.66]	1.42 [1.39, 1.44]	2.58	26/26
OLMoE	1.060 [1.058, 1.063]	2.18 [2.16, 2.20]	3.17 [3.11, 3.22]	1.44 [1.43, 1.45]	3.30	16/16
GPT-OSS	1.444 [1.433, 1.455]	2.33 [2.30, 2.35]	3.33 [3.28, 3.37]	1.47 [1.45, 1.49]	2.79	23/24
Qwen1.5	1.071 [1.068, 1.074]	1.85 [1.84, 1.87]	3.42 [3.39, 3.65]	1.94 [1.92, 2.01]	3.13	23/24
ERNIE	1.175 [1.170, 1.179]	2.67 [2.65, 2.69]	13.2 [12.8, 13.6]	4.72 [4.49, 4.81]	8.61	27/27
Moonlight	1.625 [1.612, 1.635]	3.79 [3.75, 3.81]	14.1 [13.7, 14.5]	3.96 [3.85, 4.07]	8.56	26/26

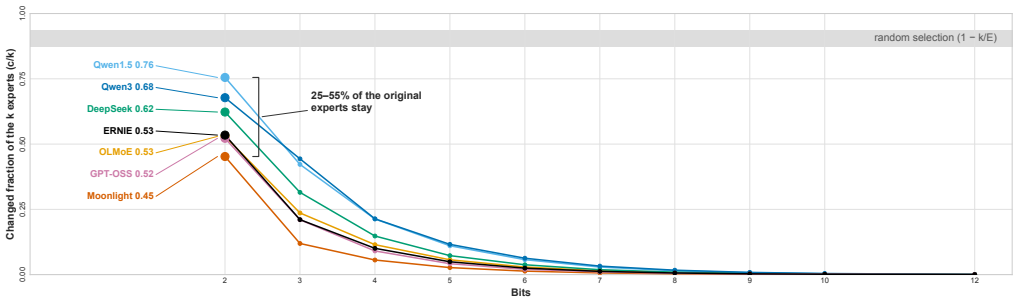

Figure 7: Fraction of the selected experts that change, c/k, versus bit width. Gray band: random selection ($1 - k/E$). At 2 bits, c/k stays below the random value.

J.4 The deviation per selection change across models and bit widths

Table 34 gives the smallest and the largest h among the seven models at each bit width. From 5 to 12 bits, the same two models, ERNIE and GPT-OSS, are at the two ends, and their ratio stays between 6.7 and 7.2. Their interquartile ranges across layers do not overlap: at 8 bits, the lower quartile of h for GPT-OSS is 4.1 times the upper quartile for ERNIE. Below 5 bits, h grows at different rates in different models, and the ratio moves to 10.2 at 4 bits, 14.0 at 3 bits, and 3.5 at 2 bits. Within each model, h at 2 bits is larger than h at 8 bits in all 191 layers (median ratio per model 5.2–23.9), and h at 3 bits is larger than at 8 bits in 189 of 191 layers.

Table 34: Smallest and largest h across the seven models at each bit width (condition (i); 95% intervals).

Bits	Smallest h	Largest h	Ratio
2	DeepSeek, 0.129	Qwen3, 0.448	3.5
3	Moonlight, 0.0166	Qwen3, 0.233	14.0
4	ERNIE, 0.00923	Qwen3, 0.0937	10.2
5	ERNIE, 0.00751	GPT-OSS, 0.0505	6.72 [6.46, 7.19]
6	ERNIE, 0.00697	GPT-OSS, 0.0475	6.81 [6.50, 7.18]
7	ERNIE, 0.00690	GPT-OSS, 0.0473	6.86 [6.55, 7.28]
8	ERNIE, 0.00666	GPT-OSS, 0.0466	6.99 [6.56, 7.40]
9	ERNIE, 0.00667	GPT-OSS, 0.0482	7.22 [6.77, 7.69]
10	ERNIE, 0.00659	GPT-OSS, 0.0465	7.05 [6.55, 7.57]
12	ERNIE, 0.00670	GPT-OSS, 0.0462	6.90 [6.22, 7.46]

At 8 bits, h is 0.00666 [0.00635, 0.00703] (ERNIE), 0.0102 [0.0100, 0.0106] (DeepSeek), 0.0106 [0.0102, 0.0109] (Moonlight), 0.0147 [0.0143, 0.0151] (Qwen1.5), 0.0159 [0.0153, 0.0169] (OLMoE), 0.0318 [0.0307, 0.0330] (Qwen3), and 0.0466 [0.0451, 0.0481] (GPT-OSS). Two models with the same change rate can therefore differ sevenfold in routing-induced deviation; the change rate gives the deviation only after h has been measured for that model.

J.5 A SATURATED CHANGE RATE AT THE MODEL OUTPUT

The same holds at the output of the whole model. In condition (ii), five settings have a layer-local change rate of 0.93–1.00 (DeepSeek, OLMoE, Qwen1.5, and Qwen3 at 3 bits, and Qwen3 at 4 bits). Their change rates are nearly equal, yet the top-1 agreement with the original model ranges from 0.33 to 0.79, and the largest output KL among them is 12.2 [11.4, 13.2] times the smallest. Selecting the settings by the change rate at the model output ($f \geq 0.95$) instead gives four settings and a ratio of 11.5.

K ADDITIONAL CONTROLS

K.1 QUANTIZING THE ATTENTION LAYERS AS WELL

Our conditions keep the attention layers in full precision (Table 2). Table 35 quantizes them as well, with the same RTN at the same bit width. The blind fraction becomes smaller in all six settings: at 8 bits from 0.850 to 0.771 (OLMoE), from 0.866 to 0.745 (DeepSeek), and from 0.886 to 0.832 (Qwen3), so the reconstruction error captures 1.5–1.9 times as large a share of the deviation. The routing-induced deviation itself does not shrink (R grows in all six settings); the rest of the deviation grows more.

Table 35: Quantizing the attention layers as well (model output). Attention alone: routers and experts in full precision. Brackets: 95% intervals.

					Attention alone	
Model	Bits	m (iv)	m (iv) + attention	Δm	f	m
OLMoE	4	0.197	0.147	-0.050 [-0.055, -0.044]	0.506	0.145
	8	0.850	0.771	-0.079 [-0.128, -0.043]	0.070	0.852
DeepSeek	4	0.189	0.103	-0.086 [-0.095, -0.078]	0.677	0.096
	8	0.866	0.745	-0.121 [-0.128, -0.115]	0.130	0.761
Qwen3	4	0.751	0.674	-0.077 [-0.096, -0.055]	0.571	0.132
	8	0.886	0.832	-0.054 [-0.062, -0.043]	0.115	0.866

Quantizing only the attention layers, with the routers and experts in full precision, also changes the expert selection: at 8 bits, 7–13% of tokens reach a different expert set (median across layers), and restoring the selection removes 76–87% of the deviation. The routing-induced deviation therefore arises even when the router weights are not quantized.

K.2 ROUNDING VARIANTS WITH THE SAME RECONSTRUCTION ERROR

We test whether the routing-induced deviation depends on the particular rounding of RTN when the reconstruction error is held fixed. Starting from RTN on the group-128 grid, we built rounding variants that change the rounding direction of 0.2–1.0% of the router weights while keeping the reconstruction error on calibration data within $\pm 1\%$ of a common target. Two variants chose the changes to lower or to raise a linear estimate of R on 32 calibration sequences; twenty others chose them at random. On the test data, the variants have the same reconstruction error of the router, O, within 0.5% and the same D_{frozen} within 5.5%.

Choosing the rounding to raise or lower R moves it by at most 1.9% at the same O (Table 36; the interval excludes 1 in 16 of 21 settings). Twenty random roundings with the same target already spread R by 1.3–6.1%, and only in one setting (DeepSeek at 4 bits) do both chosen roundings fall outside this spread. Within one grid, the routing-induced deviation is therefore not an artifact of a particular rounding.

K.3 KEEPING THE ROUTER IN FULL PRECISION

Section 1 notes that toolkits disagree on whether to quantize the router. Table 37 compares quantizing the routed experts and the routers (condition (iv)) with quantizing the routed experts alone (condition (iii)) at the same bit width. The ratio of the output KL with the routers quantized to that

Table 36: Ratio of R between the two roundings chosen to raise and to lower R, at the same O (condition (i); 95% intervals).

Model	4 bits	6 bits	8 bits
OLMoE	1.016 [1.013, 1.019]	1.002 [1.000, 1.007]	1.004 [0.999, 1.008]
Qwen1.5	1.014 [1.011, 1.017]	1.013 [1.009, 1.016]	1.006 [0.999, 1.010]
DeepSeek	1.018 [1.016, 1.021]	1.008 [1.006, 1.011]	1.003 [0.999, 1.007]
Qwen3	1.004 [1.003, 1.005]	1.008 [1.007, 1.011]	1.003 [1.001, 1.006]
Moonlight	1.019 [1.017, 1.025]	1.011 [1.008, 1.014]	1.006 [1.001, 1.011]
GPT-OSS	1.019 [1.014, 1.025]	1.013 [1.008, 1.017]	1.001 [0.997, 1.006]
ERNIE	1.018 [1.014, 1.024]	1.014 [1.010, 1.017]	1.007 [1.002, 1.011]

with the routers in full precision is 1.01–3.12 in six models across 3 to 10 bits and up to 11.6 in Qwen3 at 3 bits. The ratio depends on the model: it is 2.0–3.1 for OLMoE, Qwen1.5, and DeepSeek at 6 bits and above, and 1.01–1.03 for GPT-OSS. Its interval includes 1 in 3 of the 35 settings (GPT-OSS at 8 and 10 bits, ERNIE at 4 bits).

Table 37: Output KL with the routers quantized (condition (iv)) divided by that with the routers in full precision (condition (iii)). Super- and subscripts: 95% interval.

Model	3 bits	4 bits	6 bits	8 bits	10 bits
OLMoE	$1.571^{1.647}_{1.494}$	$2.051^{2.092}_{2.013}$	$2.197^{2.243}_{2.153}$	$2.297^{2.396}_{2.204}$	$2.226^{2.482}_{1.923}$
Qwen1.5	$2.143^{2.205}_{2.086}$	$2.309^{2.359}_{2.262}$	$2.648^{2.682}_{2.612}$	$2.853^{2.944}_{2.773}$	$3.117^{3.314}_{2.937}$
DeepSeek	$1.398^{1.421}_{1.374}$	$1.770^{1.793}_{1.745}$	$2.044^{2.095}_{1.998}$	$2.229^{2.305}_{2.151}$	$2.251^{2.492}_{2.027}$
Qwen3	$11.61^{12.13}_{11.13}$	$9.909^{11.56}_{8.614}$	$3.007^{3.196}_{2.806}$	$2.200^{2.445}_{1.986}$	$1.807^{2.027}_{1.629}$
Moonlight	$1.025^{1.031}_{1.019}$	$1.051^{1.054}_{1.046}$	$1.122^{1.139}_{1.107}$	$1.198^{1.226}_{1.172}$	$1.321^{1.490}_{1.188}$
GPT-OSS	$1.013^{1.023}_{1.003}$	$1.024^{1.031}_{1.016}$	$1.031^{1.043}_{1.018}$	$1.011^{1.025}_{0.998}$	$1.015^{1.044}_{0.989}$
ERNIE	$1.154^{1.172}_{1.138}$	$1.037^{1.085}_{0.992}$	$1.270^{1.296}_{1.242}$	$1.390^{1.433}_{1.350}$	$1.513^{1.625}_{1.418}$

Accuracy does not detect this difference either. In OLMoE on MMLU-Pro, keeping the router in full precision raises answer agreement with the original model by 17.60 points [15.15, 20.15] at 4 bits and 1.80 points [0.75, 2.85] at 8 bits, whereas the change in accuracy is indistinguishable from zero at both bit widths ($+1.45$ [-0.05, $+2.85$] and $+0.20$ [-0.15, $+0.60$]); at 8 bits, the NLL of the correct option even increases, by 0.0087 [0.0052, 0.0122].