INCMOE: CALIBRATION-FREE PRECISION REFINEMENT FOR MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

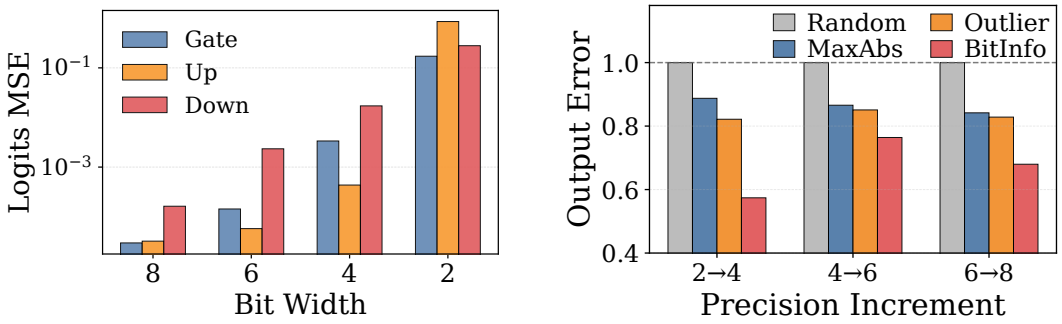
Mixture-of-Experts Models (MoE) achieve superior scaling efficiency, yet their massive parameter volume incurs prohibitive memory footprints and IO transfer overheads during serving. Mixed-precision quantization mitigates these bottlenecks by allocating non-uniform bit-widths across experts. However, existing paradigms suffer from two fundamental limitations: (i) static mixed-precision schemes enforce rigid allocations that fail to adapt to token-dependent routing dynamics, while multi-copy dynamic-precision approaches incur prohibitive memory duplication by storing multiple discrete weight copies; and (ii) conventional allocation heuristics rely on calibration activations that are vulnerable to runtime distribution shifts. To overcome these constraints, we propose INCMOE, a calibration-free, single-copy framework enabling runtime router-conditioned precision refinement. INCMOE reformulates precision from a static expert property into a dynamic trajectory of progressive refinements executed along routed paths. Specifically, each weight tensor is physically encoded once into nested, ordered bit slices comprising a compact base layer and progressively loadable increments, supporting variable precisions without parameter duplication. To identify critical increments without calibration data, we introduce BITINFO, which provides a data-free ranking score for individual bit slices directly from weight distributions to establish priority hierarchies. At runtime, a lightweight scheduler combines instantaneous gating confidences with offline priority lists, dynamically assigning prefix-consistent bit increments to active experts under a target bit budget. Experiments across three representative MoE architectures show that INCMOE approaches full-precision perplexity at an average 6-bit budget and preserves competitive downstream quality at 4 bits while strictly maintaining single-copy storage, establishing a superior accuracy-memory Pareto frontier for MoE serving.

1 INTRODUCTION

Mixture-of-Experts Models (MoE) achieve unprecedented parameter capacity with sparse per-token execution (Jiang et al., 2024). Although activating a small expert subset preserves computational throughput, retaining the entire expert reservoir and streaming routed parameters during autoregressive decoding impose prohibitive memory capacity and bandwidth bottlenecks (Rajbhandari et al., 2022). Post-training quantization (PTQ) substantially mitigates these overheads by compressing weight representations (Frantar et al., 2022; Lin et al., 2024). However, existing PTQ schemes typically enforce rigid precision profiles configured offline, using either uniform bitwidths or static mixed-precision assignments (Duanmu et al., 2025). While computationally efficient at fixed operational targets, such static assignments fundamentally fail to adapt to token-dependent routing dynamics when routing distributions shift across heterogeneous inference workloads.

Challenge. Efficient MoE serving is fundamentally governed by two orthogonal budgets: **static resident storage** for accommodating the global expert pool in memory, and **dynamic active-path precision** that dictates the weight payload transferred per token. While deploying multiple discrete model checkpoints across precisions allows runtime trade-offs, it incurs multiplied storage footprints that defeat the primary objective of compression. Although unified multi-precision representations avoid parameter duplication for dense architectures (Park et al., 2024), adapting them to MoE introduces an unresolved challenge: how to dynamically distribute a token's bit budget across routed experts and internal weight tiles without compromising single-copy storage efficiency.

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.



(a) Projection sensitivity

(b) Allocation criterion

Figure 1: Local evidence for selective refinement. (a) Gate, up, and down projections have different logits-MSE curves within an expert. (b) At matched increment budgets, BitInfo yields lower output error than the compared heuristics; each transition is normalized to random selection. End-task results are reported in Section 4.

Limitation. Conventional quantization pipelines rely heavily on calibration data to identify salient parameters (Frantar et al., 2022; Lin et al., 2024; Hu et al., 2025). When serving unknown deployment workloads, calibration sets can introduce inductive bias. Our diagnostic evaluations reveal that **calibration domain shifts severely corrupt output distributions**, frequently degrading generation quality on out-of-distribution prompts. This vulnerability motivates a robust operational paradigm: deriving precision profiles directly from intrinsic weight characteristics offline, while performing dynamic adaptation at runtime strictly using instantaneous routing signals already produced during autoregressive inference.

Our Proposal. We formulate MoE precision allocation around **progressive precision increments**. Crucially, expert gating importance and the marginal utility of refining specific weight tiles do not necessarily correlate: a heavily routed expert may contain tiles with negligible quantization residuals, whereas sub-dominant experts may harbor sensitive structures that benefit disproportionately from bit refinement. Driven by this insight, we propose INCMOE, a calibration-free framework for runtime router-conditioned precision refinement. INCMOE encodes each weight tile once into four ordered 2-bit slices of an 8-bit representation, allowing successive bit planes to dynamically refine a low-precision base without parameter redundancy. To rank candidate slices without calibration activations, our offline metric, BITINFO, scores candidate slices directly from raw weight geometries to construct priority lists. At runtime, a lightweight scheduler combines these static priorities with dynamic routing logits, allocating prefix-consistent bit increments across active experts under a prescribed budget. Across LLaMA-MoE, Mixtral, and Qwen3-MoE, INCMOE consistently delivers competitive fidelity-efficiency trade-offs under an average 4-bit active-path budget. On Mixtral, INCMOE achieves a 6-bit perplexity of 5.36, approaching the 5.34 of FP16, while compressing the physical footprint from 84.00 GiB to 44.46 GiB including allocation indices. Furthermore, single-sequence decoding at an average 4-bit budget has 24.2% lower latency than fixed INT8, with a 13.6% latency overhead over fixed INT4.

2 PROBLEM SETTING AND MOTIVATION

Consider an MoE layer l processing an input hidden state $h_l \in \mathbb{R}^d$. A parametric router assigns scalar routing probabilities to the entire expert pool and selects a sparse active subset $\mathcal{E}_l$ with $|\mathcal{E}_l| = k \ll E$. The layer output aggregates the active expert projections via normalized gating coefficients $g_{l,e}$:

$$y_l = \sum_{e \in \mathcal{E}_l} g_{l,e} f_{l,e}(h_l), \qquad \sum_{e \in \mathcal{E}_l} g_{l,e} = 1. \tag{1}$$

We quantize the parameter-dense feed-forward projections within each expert, namely $x \in \mathcal{X} = \{\text{gate}, \text{up}, \text{down}\}$. Non-expert modules, including self-attention blocks, token embeddings, and router networks, retain their native precisions following standard practice (Frantar et al., 2022; Lin et al., 2024). While activation sparsity allows inactive experts to bypass execution, storing the full expert reservoir still dominates memory footprint. Standard static and dynamic quantization

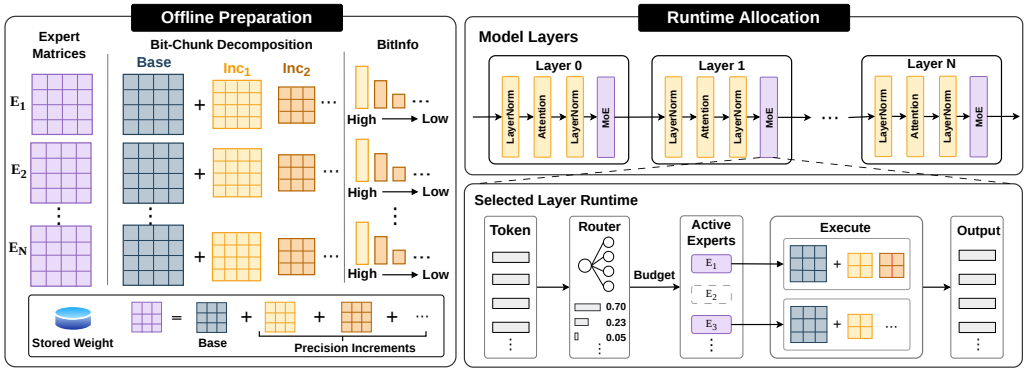


Figure 2: IncMoE. Offline, weights are encoded into ordered chunks and profiled with BitInfo. At runtime, the router identifies active experts, the allocator assigns increment quotas, and selected chunks refine each projection's base computation. All operating budgets share the same stored expert representation.

methods exploit this activation sparsity to compress the resident pool; INCMOE advances beyond static allocations by dynamically modulating precision *within* the activated expert paths.

Observation 1. Quantization vulnerability exhibits multi-dimensional heterogeneity, showing that **coarse expert-level precision allocation is suboptimal**. As Figure 1a and Appendix C show, sensitivity diverges across projection matrices within the same expert. This sensitivity varies non-linearly across precision levels: the relative error gap between up and down projections shifts between 2 bits and 4 bits. Static scalar rankings over entire experts therefore cannot identify where an added bit increment yields the highest marginal gain. In addition, increment energy is concentrated in local weight tiles (Appendix Figure 5). Under matched bit budgets, allocating increments with our BITINFO metric achieves lower output error than heuristic alternatives (Figure 1b). This localized concentration requires scheduling bit refinements at sub-matrix tile granularity rather than treating entire expert blocks uniformly.

Observation 2. Data-driven calibrations introduce domain-specific bias, showing that **calibration-dependent precision assignment remains vulnerable to inference distribution shifts**. Holding the evaluation domain constant, alternating the calibration domain increases output KL divergence by up to $3.16\times$ for AWQ and $2.64\times$ for SmoothQuant relative to the best tested calibration source for that evaluation domain (Appendix C). The optimal calibration distribution also varies across quantization algorithms. This brittleness indicates that activation-based importance profiling risks over-fitting to calibration corpora at the expense of unseen deployment workloads. We therefore adopt a decoupled approach: precomputing structural sensitivity profiles strictly from intrinsic weight geometries offline, while dynamically orchestrating bit allocations at runtime via routing confidences produced during decoding. Section 4.3 evaluates this source-versus-unseen domain trade-off directly.

3 INCMOE FRAMEWORK

Figure 2 summarizes the offline representation and runtime allocation. Let $W^x_{l,e,r}$ denote tile r in projection x of expert e at layer l. The method keeps the stored tiles fixed and selects their refinement depth for each token t.

3.1 SINGLE REPRESENTATION WITH NESTED BIT-PLANE PRECISION

For each weight tile $W_{l,e,x,r}$, a calibration-free range search determines an optimal affine quantizer by minimizing weight Frobenius error:

$$s^*_{l,e,r,x} = \arg\min_{s\in\mathcal{S}^x_{l,e,r}} \left\| W^x_{l,e,r} - D_s\big(Q_s(W^x_{l,e,r})\big)\right\|_F^2 . \tag{2}$$

where candidate scaling sets $\mathcal{S}_{l,e,x,r}$ are derived by contracting the absolute-maximum parameter range. Both quantizer Q_s and dequantizer D_s incorporate the optimal zero-point ζ corresponding to

candidate range s. This optimization yields an unsigned 8-bit quantized integer tensor Q, scale s, and zero-point ζ strictly from raw weights without activation data.

Suppressing structural indices for clarity, the quantized integer tensor Q decomposes elementwise into four nested 2-bit chunks:

$$Q = \sum_{i=0}^{3} 2^{2(3-i)} C^i, \qquad C^i \in \{0, 1, 2, 3\}. \tag{3}$$

This decomposition establishes a 2-bit base representation ($b_0 = 2$) augmented by three successive 2-bit increments:

$$W^{\text{base}} = s(2^6 C^0 - \zeta), \qquad \Delta W^m = s\, 2^{2(3-m)} C^m, \quad m = 1, 2, 3. \tag{4}$$

The zero-point ζ is folded exclusively into the base computation, ensuring that subsequent refinement stages require only unsigned residual additions. Summing all components reconstructs the dequantized 8-bit tensor without loss. Choosing 2-bit granularity forms a regular lattice of operational states $\{2, 4, 6, 8\}$ bits, balancing hardware SIMD byte-packing constraints with fine-grained scheduling flexibility.

During decoding, binary indicators $z^m_{l,e,x,r}(t) \in \{0, 1\}$ govern chunk activation. The materialized tile weight and effective bitwidth at token step t are given by:

$$\widetilde{W}^x_{l,e,r}(t) = W^{x,\text{base}}_{l,e,r} + \sum_{m=1}^{3} z^{x,m}_{l,e,r}(t) \Delta W^{x,m}_{l,e,r}, \tag{5}$$

$$b_{l,e,x,r}(t) = 2 + 2 \sum_{m=1}^{3} z^{x,m}_{l,e,r}(t), \qquad z^{x,m+1}_{l,e,r}(t) \le z^{x,m}_{l,e,r}(t). \tag{6}$$

The prefix-monotonicity constraint in Equation (6) enforces contiguous loading, preventing higher-order increments from activating without their preceding bit planes. Crucially, all four chunks reside continuously within single-copy storage; executing under an average 4-bit budget avoids maintaining multiple discrete checkpoints.

A b-bit prefix corresponds to zero-padded bit truncation $W^{[b]} = s(2^{8-b}\lfloor Q/2^{8-b} \rfloor - \zeta)$. For $s > 0$, the entrywise approximation error relative to full 8-bit dequantization satisfies:

$$\|D_s(Q) - W^{[b]}\|_{\max} \le s(2^{8-b} - 1), \qquad b \in \{2, 4, 6, 8\}, \tag{7}$$

where $\|\cdot\|_{\max}$ denotes the maximum absolute entry. Because the truncated integer residue is strictly bounded by $2^{8-b} - 1$, each progressive chunk monotonically tightens the reconstruction ceiling. Overall weight reconstruction error combines this prefix bound with the initial clipping and rounding residuals defined in Equation (2).

3.2 Weight-only increment priorities

BitInfo combines increment energy with a chunk-symbol score:

$$\mathcal{I}^{x,m}_{l,e,r} = \mathcal{G}^{x,m}_{l,e,r} \mathcal{B}^{x,m}_{l,e,r}, \qquad \mathcal{G}^{x,m}_{l,e,r} = \|\Delta W^{x,m}_{l,e,r}\|^2_F. \tag{8}$$

For the symbols c_j in a tile's chunk,

$$\mathcal{B}^{x,m}_{l,e,r} = -\frac{1}{|r|} \sum_{j \in r} \log p_{x,m}(c_j), \tag{9}$$

where $p_{x,m}$ is the empirical distribution pooled across layers and experts for projection type x and stage m. Pooling supplies a common reference for comparing tiles and avoids estimating a separate reference from each small tile. Equation (9) is mean surprisal under that reference, or cross-entropy, rather than the entropy of the tile's own histogram. A rare repeated symbol can therefore receive a high score despite having zero within-tile entropy.

The product favors increments with both numerical magnitude and less common symbol content. It is not the exact reduction in reconstruction error: for residual $R = W - \widetilde{W}$, adding ΔW reduces

squared error by $2\langle R, \Delta W\rangle - \|\Delta W\|_F^2$, which also depends on R. Downstream error additionally depends on the input activations. We use BitInfo as an inexpensive data-free ranking criterion and test its factors separately in Section 4.5. Global pooling and the product form are empirical choices.

For each expert projection, we compile a list $\Pi_{l,e}^x$ of (r, m) pairs offline. Initially only stage 1 of each tile is eligible. Repeatedly selecting the eligible pair with highest BitInfo, then exposing that tile's next stage, produces a dependency-respecting order; ties use a fixed index order. Every prefix of $\Pi_{l,e}^x$ is therefore feasible. This greedy compilation avoids online sorting, but need not maximize total utility over all feasible selections. Per-increment scores can be discarded after compilation; runtime retains packed indices and the expert/projection summary statistics used below.

3.3 Router-conditioned budget allocation

Let $\mathcal{R}_l(t)$ contain the equal-sized tiles across all three projections of the active experts, and let $N_l(t) = |\mathcal{R}_l(t)|$. For target average precision $\bar{b} \in [2, 8]$, we select

$$K_l(t) = \mathrm{round}\left(\frac{N_l(t)(\bar{b} - 2)}{2}\right) \tag{10}$$

increments. The realized precision is $\widetilde{b}_l(t) = 2 + 2K_l(t)/N_l(t)$, so $|\widetilde{b}_l(t) - \bar{b}| \le 1/N_l(t)$. This is an active expert-weight payload budget, excluding metadata and non-expert computation. For unequal tile sizes, both cost and average precision must be weighted by parameter count; an unweighted quota would not enforce the same budget.

Budgeted refinement as a selection problem. BitInfo supplies an additive proxy for comparing feasible allocations. Writing $a = (e, x, r)$ for an active tile and abbreviating its score and selection by $\mathcal{I}_a^m$ and z_a^m, the corresponding reference problem is

$$\begin{aligned} \max_{z \in \{0,1\}^{3N_l(t)}} \quad & \sum_{a=(e,x,r)\in\mathcal{R}_l(t)} \sum_{m=1}^{3} (g_{l,e}(t) + \epsilon)\mathcal{I}_a^m z_a^m \\ \text{subject to} \quad & \sum_{a\in\mathcal{R}_l(t)} \sum_{m=1}^{3} z_a^m = K_l(t), \qquad z_a^{m+1} \le z_a^m \quad (m = 1, 2). \end{aligned} \tag{11}$$

Each tile contributes one of four prefix depths, coupling its increment decisions. Exact dynamic programming over tiles and quota takes $O(N_l(t)K_l(t))$ time (Appendix A.3), making repeated tile-level optimization costly during decoding. IncMoE instead uses aggregate scores to apportion quotas and precompiled lists to select their contents. This hierarchical rule enforces the same constraints while approximating the utility objective; it carries no optimality guarantee for that objective or for downstream loss.

The expert allocation weight is

$$H_{l,e}(t) = \big(g_{l,e}(t) + \epsilon\big)\big(\bar{\mathcal{I}}_{l,e} + \epsilon\big), \qquad \epsilon > 0, \tag{12}$$

where $\bar{\mathcal{I}}_{l,e}$ averages the candidate increment scores in the expert. Both factors increase the expert's proportional share when the other is held fixed. Within each expert, projection shares are proportional to $\bar{\mathcal{I}}_{l,e}^x + \epsilon$. The constant ϵ prevents a degenerate zero total weight.

Both stages use capacity-constrained proportional allocation. A component is saturated if its share exceeds its number of available increments; the remaining quota is redistributed among unsaturated components. Largest-remainder rounding gives integer quotas without changing their sum. The selected increments are then

$$\mathcal{S}_{l,e}^x(t) = \mathrm{Prefix}\big(\Pi_{l,e}^x, K_{l,e}^x(t)\big), \qquad \sum_{e\in\mathcal{E}_l(t)} \sum_{x\in\mathcal{X}} K_{l,e}^x(t) = K_l(t). \tag{13}$$

This rule conserves the layer quota and respects all tile dependencies (Appendix A). Gate scores influence allocation across active experts; BitInfo supplies the expert/projection summaries and local order. Runtime allocation operates on the k selected experts and three projections per expert. Fetching indices and executing the selected increments still scale with their number; the full runtime is not constant-cost.

Algorithm 1 Router-conditioned refinement at layer l for token t

Require: Target $\bar{b}$; offline lists $\Pi^x_{l,e}$, score means, and capacities
1: Obtain active experts $\mathcal{E}_l(t)$ and normalized weights $g_{l,e}(t)$
2: Compute $K_l(t)$ from the active tile count using Equation (10)
3: $K_{l,e} \leftarrow \mathrm{Apportion}(K_l, \{H_{l,e}\}, \{c_{l,e}\})$
4: **for** each $e \in \mathcal{E}_l(t)$ **do**
5: $\quad \{K^x_{l,e}\}_{x\in\mathcal{X}} \leftarrow \mathrm{Apportion}(K_{l,e}, \{\bar{\mathcal{I}}^x_{l,e} + \epsilon\}_{x\in\mathcal{X}}, \{|\Pi^x_{l,e}|\}_{x\in\mathcal{X}})$
6: $\quad \mathcal{S}^x_{l,e} \leftarrow \Pi^x_{l,e}[1 : K^x_{l,e}]$ for each projection x
7: **end for**
8: Execute gate/up projections, the nonlinear product, and down projection

Algorithm 1 summarizes the online path. Here $c_{l,e} = \sum_x |\Pi^x_{l,e}|$, and $\mathrm{Apportion}$ denotes the capped allocation and integer rounding above; a zero quota selects an empty prefix. At $\bar{b} = 2$ the algorithm executes only the base, whereas at $\bar{b} = 8$ all lists are exhausted and the full dequantized 8-bit expert is recovered. Intermediate budgets redistribute increments without changing the expert identities chosen by the router.

Preprocessing and control cost. For a projection with R tiles, heap-based priority compilation visits its $3R$ increments in $O(R \log R)$ time and stores $O(R)$ indices. The router selects the active set before allocation. A straightforward implementation of expert apportionment takes at most k saturation passes over k experts, or $O(k^2)$ work; projection apportionment has only three components. Reading the chosen prefixes adds $O(K_l(t))$ index accesses. These costs explain which work is moved offline and which still scales with runtime refinement; weight decoding and projection arithmetic remain additional costs.

3.4 Incremental execution

For each linear projection, with tile increments understood as zero outside their matrix region,

$$h\widetilde{W}^x_{l,e}(t) = hW^{x,\mathrm{base}}_{l,e} + \sum_{(r,m)\in\mathcal{S}^x_{l,e}(t)} h\Delta W^{x,m}_{l,e,r}. \tag{14}$$

The implementation stores chunks in a structured packed layout and groups selected increments by stage for tile-kernel dispatch. Fused Triton/CUDA execution decodes and accumulates their contributions without materializing a token-specific full weight matrix. Equation (14) applies separately to each projection: gate and up results are assembled before the activation and elementwise product, and the down projection operates on that intermediate. It does not linearize the expert's nonlinear computation.

Tokens routed to the same expert share priority lists but may request different prefix lengths. Aligning to the longest prefix can exceed individual budgets; grouping compatible prefixes trades this inflation against dispatch cost and utilization. Our timing evaluation covers batch-size-one decoding, leaving these batching trade-offs unmeasured.

4 Experiments

4.1 Evaluation setup

We evaluate LLaMA-MoE-3.5B (Zhu et al., 2024), Mixtral-8×7B (Jiang et al., 2024), and Qwen3-30B-A3B (Yang et al., 2025). The latter uses 128 routed experts with top-8 routing, providing a different routing topology. The comparison includes FP16, MoQE (Kim et al., 2023), GPTQ (Frantar et al., 2022), MxMoE (Duanmu et al., 2025), and MoEQuant (Hu et al., 2025). Main-table quantization is restricted to expert weights. IncMoE uses a 2-bit base and average targets $\bar{b} \in \{4, 5, 6\}$, all drawn from the same resident 8-bit representation.

We report WikiText perplexity (Merity et al., 2016), accuracy on BoolQ and MMLU (Clark et al., 2019; Hendrycks et al., 2020), normalized accuracy on HellaSwag (Zellers et al., 2019), GSM8K exact

Table 1: Quality at a 4-bit expert-weight execution budget, with FP16 as the full-precision reference. For IncMoE, bits denote the average *executed* precision, not resident storage. Calib. reports the number of calibration sequences × tokens per sequence. TQA is TruthfulQA-Gen ROUGE-L.

Method	Bits	Calib.	PPL ↓	BoolQ	HellaS.	MMLU	GSM8K	TQA	MBPP
LLaMA-MoE-3.5B									
FP16	16	–	10.48	72.56	54.20	27.44	4.32	24.96	8.00
GPTQ	4	128×2K	10.69	72.93	53.32	26.71	4.01	25.33	6.80
MxMoE	4	128×4K	11.51	69.26	52.13	25.62	3.18	24.60	5.40
MoEQuant	4	128×512	10.71	71.19	53.36	26.07	3.41	24.84	7.00
MoQE	4	0	10.92	65.81	43.50	24.47	3.71	24.84	5.40
IncMoE	**4**	**0**	**10.75**	**71.74**	**52.62**	**25.89**	**4.09**	**26.19**	**7.00**
Mixtral-8×7B									
FP16	16	–	5.34	86.02	84.26	68.40	59.96	43.45	48.40
GPTQ	4	128×2K	5.60	85.54	84.11	67.48	55.04	41.00	46.60
MxMoE	4	128×4K	5.71	85.56	83.42	67.62	55.48	41.25	45.40
MoEQuant	4	128×512	5.79	84.49	83.88	67.51	56.17	41.24	46.60
MoQE	4	0	8.26	82.14	80.06	60.32	29.72	28.15	30.20
IncMoE	**4**	**0**	**5.68**	**85.44**	**83.85**	**67.14**	**56.33**	**41.49**	**47.00**
Qwen3-30B-A3B									
FP16	16	–	11.57	88.53	59.55	77.70	85.82	39.05	71.80
GPTQ	4	128×2K	11.93	88.38	58.73	76.99	81.65	38.43	68.20
MxMoE	4	128×4K	11.87	88.50	58.93	77.27	87.49	41.62	69.60
MoEQuant	4	128×512	11.83	88.38	58.91	76.76	89.31	39.53	69.20
MoQE	4	0	21.52	87.19	50.16	70.60	84.84	40.64	63.60
IncMoE	**4**	**0**	**12.03**	**88.65**	**58.97**	**77.12**	**87.87**	**44.92**	**70.00**

match (Cobbe et al., 2021), TruthfulQA-Gen ROUGE-L (Lin et al., 2022), and MBPP pass@1 (Austin et al., 2021). TruthfulQA ROUGE-L measures reference overlap, not a direct truthfulness judgment. All task scores are percentages; their unweighted mean is descriptive because the metrics differ. Comparisons use the common evaluation protocol within each model, rather than combining absolute scores from different papers. Experiments use NVIDIA H20 GPUs; latency measures single-sequence autoregressive decoding including online allocation, increment indexing/dispatch, computation, and accumulation. Results are point estimates without multi-seed uncertainty, so small differences should not be read as statistically established improvements.

4.2 Quality and precision scaling

Table 1 and Table 2 evaluate downstream task performance and dynamic fidelity recovery across LLaMA-MoE, Mixtral, and Qwen3-MoE.

Task performance at 4-bit budget. Despite operating without calibration activations, IncMoE maintains parity with calibration-dependent static baselines. On Mixtral, IncMoE achieves the highest scores among all 4-bit methods on GSM8K, TruthfulQA, and MBPP, leading the six-task average at 63.54 compared to 63.30 for GPTQ, 63.12 for MxMoE, and 63.32 for MoEQuant. On LLaMA-MoE, IncMoE attains top scores on GSM8K and TruthfulQA. On the larger expert pool of Qwen3, IncMoE remains competitive, achieving the highest point estimates on BoolQ, HellaSwag, TruthfulQA, and MBPP among the evaluated 4-bit methods.

Dynamic precision scaling. As shown in Table 2, IncMoE enables progressive quality recovery by scaling the active-path budget without altering the stored weights. Increasing the budget from 4 to 6 bits substantially narrows the relative perplexity gap to FP16: from 2.6% down to 0.8% on LLaMA-MoE, and from 6.4% down to 0.4% on Mixtral (5.36 versus 5.34 in FP16). This establishes a vital operational advantage: while conventional PTQ commits to a single static operating point, IncMoE allows serving systems to seamlessly trade execution throughput for higher fidelity by activating deeper bit slices from the same resident representation.

Table 2: Precision scaling from one resident representation. Mean is the unweighted average of the six downstream scores in Table 1; it is not an additional benchmark.

	LLaMA-MoE-3.5B			Mixtral-8×7B		
Setting	PPL ↓	GSM8K	Mean	PPL ↓	GSM8K	Mean
FP16	10.48	4.32	31.91	5.34	59.96	65.08
IncMoE, 4 bits	10.75	4.09	31.26	5.68	56.33	63.54
IncMoE, 5 bits	10.64	4.32	31.72	5.49	57.61	64.31
IncMoE, 6 bits	10.56	4.77	32.24	5.36	59.13	65.04

Table 3: Fixed-source domain-shift evaluation: KL divergence ↓. Calibrated baselines use 128 GSM8K samples. Bold identifies the lowest value in each row. The source domain is excluded from the final mean.

Evaluation domain	AWQ	GPTQ	SmoothQuant	IncMoE-4
GSM8K (source)	0.384	0.429	**0.316**	0.517
WikiText	3.638	4.243	3.163	**2.863**
MBPP	1.890	2.217	1.650	**1.500**
HumanEval	1.298	1.518	1.131	**1.037**
HellaSwag	1.985	2.319	1.731	**1.570**
WinoGrande	0.069	**0.025**	0.032	0.052
Mean, unseen domains	1.776	2.064	1.541	**1.404**

4.3 Calibration-source dependence

To test a fixed-source deployment scenario, AWQ, GPTQ, and SmoothQuant are calibrated once on 128 GSM8K samples and evaluated on the source and five unseen domains. IncMoE uses the same calibration-free representation throughout. Table 3 reports output-distribution KL divergence, retaining the source-domain and WinoGrande results as well as the unseen-domain mean.

IncMoE has lower KL on WikiText, MBPP, HumanEval, and HellaSwag, but higher KL than all three calibrated methods on GSM8K. On WinoGrande, GPTQ and SmoothQuant are better. Mean unseen-domain KL is 1.404, an 8.9% reduction relative to SmoothQuant, 20.9% relative to AWQ, and 32.0% relative to GPTQ. These averages weight domains equally despite their different KL scales. The result supports a particular trade-off under this limited single-source calibration protocol. It does not establish superiority to larger, mixed-domain, expert-balanced, or target-matched calibration sets.

4.4 Storage and single-sequence decoding

Table 4 evaluates the trade-off between resident expert storage and runtime decoding latency on Mixtral. The complete IncMoE representation (weights, quantization metadata, and packed priority indices) consumes 44.46 GiB, reducing the 84.00 GiB FP16 expert pool by 47.1% and outperforming discrete multi-copy storage by 53.7%. During single-sequence decoding, IncMoE under an average 4-bit budget achieves 37.62 ms/token, with 24.2% lower latency than fixed INT8 (49.66 ms/token) and a 4.49 ms/token end-to-end latency overhead relative to fixed INT4 (33.13 ms/token). Scaling the operational target to 5 and 6 bits yields intermediate latencies of 41.75 and 45.88 ms/token, demonstrating that IncMoE provides fine-grained, dynamic latency scaling from a single resident deployment without redundant model checkpoints.

4.5 Allocation ablations

Table 5 evaluates allocation components and tile selection criteria on Mixtral using TruthfulQA-Gen under matched bit budgets. In the component ablation, removing either dynamic routing confidences (BitInfo-only) or intrinsic weight sensitivity (router-only) degrades ROUGE-L from 41.5 to 40.5 and 40.0, respectively, demonstrating that both signals are essential. Furthermore, the joint criterion (energy × information) yields higher point estimates than the isolated factors and heuristic baselines

Table 4: Resident expert storage and single-sequence H20 decoding. The storage budget is unchanged across IncMoE runtime targets. Latency ratios use fixed INT8 as the reference.

Expert representation	GiB	/ FP16
FP16	84.00	100.0%
Separate 4/6/8-bit copies	95.98	114.3%
Separate 2/4/6/8-bit copies	106.97	127.3%
IncMoE weights + scales/zeros	42.49	50.6%
Packed priority indices	1.97	2.3%
IncMoE total	44.46	52.9%

Execution	ms/token	/ INT8
Fixed INT4	33.13	0.67
Fixed INT8	49.66	1.00
IncMoE-4	37.62	0.76
IncMoE-5	41.75	0.84
IncMoE-6	45.88	0.92

Table 5: Mixtral allocation ablations on a 50% TruthfulQA-Gen subset. R-1/2/L denote ROUGE scores. Panels report separate component and criterion comparisons; BitInfo-only and full router-conditioned allocation are distinct settings.

Variant	BLEU	R-1	R-2	R-L	Mean
Allocator components					
Full IncMoE	45.5	42.3	37.8	41.5	41.8
BitInfo-only	44.5	41.5	37.0	40.5	40.9
Router-only	44.0	40.9	36.5	40.0	40.4
Tile-priority criterion					
Magnitude / MaxAbs	43.3	40.3	35.9	39.5	39.8
Energy only	43.9	40.9	36.4	40.0	40.3
Information only	44.0	41.0	36.5	40.1	40.4
BitInfo: energy $\times$ information	44.5	41.5	37.0	40.5	40.9
Uniform	43.8	40.7	36.2	39.8	40.1
Random	43.0	39.8	35.5	39.0	39.3

(e.g., magnitude and random), supporting the benefit of combining increment energy with weight-derived symbol information.

5 Related Work

GPTQ, AWQ, and SmoothQuant improve low-bit quantization using activation-aware reconstruction or rescaling (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023). For MoE, MoQE studies quantization robustness, while MxMoE, MoEQuant, AlphaQ, and GEMQ explore mixed-precision assignment across experts (Kim et al., 2023; Duanmu et al., 2025; Hu et al., 2025; Yang et al., 2026; Deng et al., 2026). IncMoE instead performs calibration-free precision refinement at runtime, allocating precision within token-selected experts down to projection and tile granularity.

Multi-precision representations also support multiple operating points from shared weights. AdaBits and Any-Precision enable multiple bit-widths, DP-LLM dynamically adjusts layer precision, and SliceMoE uses bit-sliced experts for on-demand precision (Jin et al., 2020; Park et al., 2024; Kwon et al., 2025; Choi et al., 2025). IncMoE further distributes each token's active-path budget across experts, projections, and local tiles through prefix-consistent increments from a single resident representation.

6 Conclusion

IncMoE separates stored expert capacity from precision executed along each token's route. Ordered chunks, weight-only BitInfo priorities, and router-conditioned quotas enable several operating budgets from one representation. Experiments quantify task-dependent quality, storage savings, and the runtime cost relative to fixed precision. The method offers a practical trade-off when calibration-free preparation and runtime precision control are deployment requirements.

AI USE STATEMENT

Generative AI assisted with manuscript organization, language editing, interpretation of existing results, reference checks, and LaTeX preparation. It also assisted with the reference allocation formulation, algorithm description, and the reconstruction, feasibility, rounding, and complexity arguments in Section 3 and Appendix A. Experimental values were drawn from the existing manuscript and evaluation records; no new experimental measurements were generated by these tools.

REPRODUCIBILITY STATEMENT

Section 3 defines the representation, scores, and runtime quotas; Algorithm 1 summarizes the online procedure. Appendix A specifies dependency-aware ordering and quota apportionment, and Appendix B provides the complete precision-scaling results. Section 4 states the evaluated models, metrics, quantization scope, and timing scope. The numerical results are reported without run-to-run uncertainty; the evidence supports the measured configurations rather than unrestricted serving workloads.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yuseon Choi, Sangjin Kim, Jungjun Oh, Gwangtae Park, Byeongcheol Kim, and Hoi-Jun Yoo. SliceMoE: Bit-sliced expert caching under miss-rate constraints for efficient moe inference. *arXiv preprint arXiv:2512.12990*, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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. *arXiv preprint arXiv:2605.23078*, 2026.

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

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Qing Jin, Linjie Yang, and Zhenyu Liao. Adabits: Neural network quantization with adaptive bit-widths. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2146–2156, 2020.

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

Sangwoo Kwon, Seong Hoon Seo, Jae W. Lee, and Yeonhong Park. DP-LLM: Runtime model adaptation with dynamic layer-wise precision assignment. In *Advances in Neural Information Processing Systems*, 2025.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

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

Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W Lee. Any-precision LLM: Low-cost deployment of multiple, different-sized llms. *arXiv preprint arXiv:2402.10517*, 2024.

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

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

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

Wanqi Yang, Yuexiao Ma, Alexander Conzelmann, Xiawu Zheng, Michael W Mahoney, T Konstantin Rusch, and Shiwei Liu. AlphaQ: Calibration-free bit allocation for mixture-of-experts quantization. *arXiv preprint arXiv:2606.04980*, 2026.

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

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*, 2024.

A Allocation Details and Guarantees

A.1 Compiling a feasible priority order

For each expert projection with R tiles, insert each tile's first optional increment into a max-heap keyed by BitInfo, breaking ties lexicographically by (r, m). Pop the largest eligible item, append it to Π, and insert the next stage of the same tile if one exists. Continue until all $3R$ items have been emitted. Only weights enter this construction. Compilation takes $O(R \log R)$ time because each tile has three optional stages and the heap contains at most R items. Storing all emitted pairs requires $O(R)$ indices. Token-time selection reads a prefix of this fixed sequence rather than sorting tiles again.

Prefix feasibility. When (r, m) is appended, either $m = 1$ or $(r, m-1)$ has already been appended. By induction, every predecessor of every selected item occurs earlier in Π. Truncating Π at any length therefore preserves the tile dependencies. This proves feasibility, not optimality: an initially low-scoring increment may unlock a high-scoring successor that the greedy procedure reaches late.

A.2 Capacity-constrained apportionment

Let a set of components have positive weights a_j, integer capacities c_j, and an integer quota K satisfying $0 \le K \le \sum_j c_j$. For expert allocation, $a_e = H_{l,e}$ and c_e is the expert's total number of optional increments; for projection allocation, $a_x = \bar{\mathcal{I}}^x_{l,e} + \epsilon$ and $c_x = |\Pi^x_{l,e}|$.

Start with all components unsaturated and remaining quota K. Assign each unsaturated component the provisional real share $u_j = K a_j / \sum_i a_i$, with the sum taken over the unsaturated set. If $u_j \ge c_j$, fix that component's share at capacity, subtract its capacity from the remaining quota, and remove it from the set. Recompute the remaining shares until none exceeds its capacity. At least one component is removed on each saturation pass, so the procedure terminates. Take floors of the final shares and give each remaining unit to a component with available capacity in descending order of fractional remainder, with fixed-index tie-breaking.

Quota conservation and rounding. The real-valued shares sum to the original quota. Their floored sum differs from that quota by an integer smaller than the number of noninteger shares. Those shares have capacity for an additional unit because capacities are integers. Largest-remainder rounding therefore yields integers K_j with $0 \le K_j \le c_j$ and $\sum_j K_j = K$. Applying this at the expert and projection levels proves Equation (13). Since the total candidate capacity is $3N_l(t)$ and $\bar{b} \in [2, 8]$, the rounded quota in Equation (10) is feasible. Finally,

$$\left|\widetilde{b}_l(t) - \bar{b}\right| = \frac{2}{N_l(t)} \left|\mathrm{round}\left(\frac{N_l(t)(\bar{b}-2)}{2}\right) - \frac{N_l(t)(\bar{b}-2)}{2}\right| \le \frac{1}{N_l(t)}.$$

Nearest-integer rounding may slightly exceed the target. If a hard upper bound is required instead, replacing rounding by a floor gives $\widetilde{b}_l(t) \le \bar{b}$ with shortfall less than $2/N_l(t)$; this is a variant of the stated allocation rule, not an additional evaluated configuration.

A.3 Exact solution of the reference utility problem

Equation (11) is a reference objective for explaining the allocation trade-off, not an optimization performed by the implementation. Index the active tiles by $j = 1, \ldots, N_l(t)$. Let $u_j(d)$ be the router-weighted sum of the first d optional increment scores of tile j, with $u_j(0) = 0$. Define $F_j(q)$ as the largest utility obtainable from the first j tiles using exactly q increments. The recurrence

$$F_j(q) = \max_{0 \le d \le \min(3,q)} \{F_{j-1}(q-d) + u_j(d)\}$$

with $F_0(0) = 0$ and all other initially infeasible states set to $-\infty$ enumerates the four feasible prefix depths of each tile. There are $O(N_l(t)K_l(t))$ states and at most four transitions per state; storing only adjacent rows uses $O(K_l(t))$ working memory when computing the objective value. This establishes the stated time cost. IncMoE's proportional quotas and offline greedy lists avoid this per-token dynamic program, and can select a different feasible solution.

A.4 What BitInfo does and does not estimate

With $R = W - \widetilde{W}$ and a candidate increment ΔW, the exact squared reconstruction-error improvement is

$$\|R\|_F^2 - \|R - \Delta W\|_F^2 = 2\langle R, \Delta W\rangle - \|\Delta W\|_F^2.$$

Increment energy alone omits the residual alignment. For a column input vector h and residual matrix R, the expected squared output error of a linear map is

$$\mathbb{E}\|Rh\|_2^2 = \mathrm{tr}\big(R\Sigma_h R^\top\big), \qquad \Sigma_h = \mathbb{E}[hh^\top].$$

A weight-only score does not observe Σ_h. Router weights also do not identify this second moment or the local sensitivity of nonlinear expert activations. These identities explain why the allocation criterion must be judged empirically rather than treated as an exact downstream-error model.

The empirical reference in Equation (9) is evaluated only on symbols present in the pooled weights, so every observed symbol has positive empirical probability. Unobserved symbols never enter that sum. Changing the logarithm base rescales all BitInfo values by a common positive factor and leaves the offline order unchanged; because Equation (12) adds ϵ, exact scale invariance of the runtime allocation would additionally require rescaling ϵ consistently.

B Full Precision-Scaling Results

Table 6 retains all reported task scores for the three runtime budgets. Each tile uses one of the four prefix precisions, while a model's active-path average can be 5 bits through a mixture of tile states. The representation and its resident storage do not change across rows.

Table 6: Complete precision-scaling results. Metric definitions match Table 1.

Setting	Bits	PPL ↓	BoolQ	HellaS.	MMLU	GSM8K	TQA	MBPP
LLaMA-MoE-3.5B								
FP16	16	10.48	72.56	54.20	27.44	4.32	24.96	8.00
IncMoE	4	10.75	71.74	52.62	25.89	4.09	26.19	7.00
IncMoE	5	10.64	72.08	53.16	27.09	4.32	26.25	7.40
IncMoE	6	10.56	73.39	54.09	27.14	4.77	26.25	7.80
Mixtral-8×7B								
FP16	16	5.34	86.02	84.26	68.40	59.96	43.45	48.40
IncMoE	4	5.68	85.44	83.85	67.14	56.33	41.49	47.00
IncMoE	5	5.49	85.69	83.96	67.94	57.61	41.86	48.80
IncMoE	6	5.36	85.87	84.09	68.24	59.13	43.08	49.80

C Additional Quantization Diagnostics

Figure 3 shows variation across sampled inputs and active expert paths under a 4-bit setting. It complements the within-expert projection diagnostic in Figure 1. Such variation motivates conditioning on the route but does not itself prove that router weights predict quantization sensitivity.

Figure 4 varies calibration source while holding each evaluation domain fixed. Entries are normalized by the smallest KL value among the tested calibration sources for the same method and evaluation domain. General-domain calibration is best in this AWQ diagnostic, whereas commonsense calibration is best for SmoothQuant. These normalized ratios are separate from the absolute KL values in Table 3 and should not be compared numerically across the two experiments.

Figure 5 reports the concentration of weight-derived increment energy. The top 20% of ranked tiles account for approximately 42–47% of the total across the three stages. This is a weight-space statistic; Figure 1b separately evaluates the output error of the selection criteria.

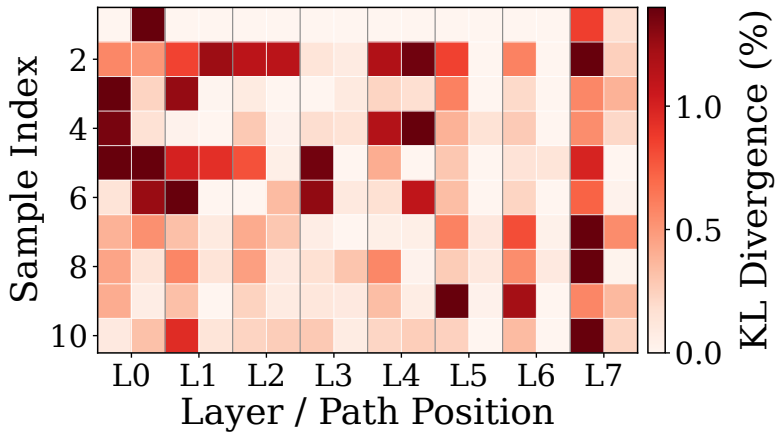
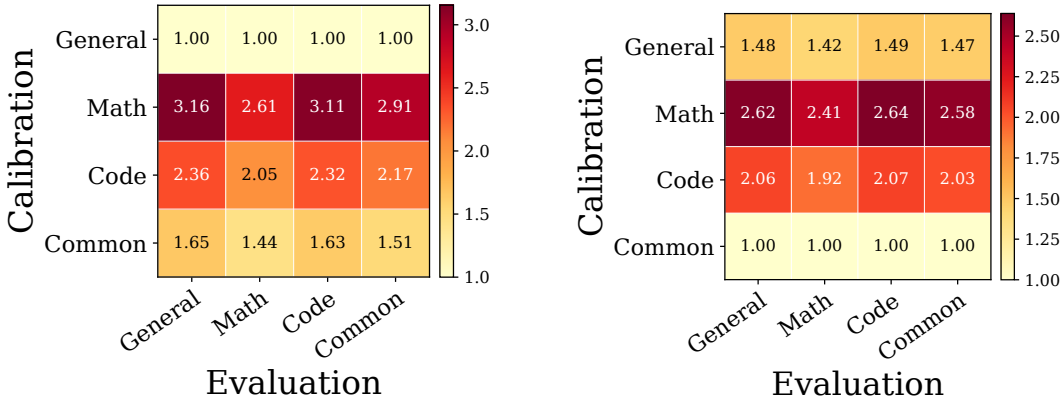


Figure 3: Input-dependent variation in the active-path sensitivity diagnostic. Columns correspond to the displayed layer/path positions L0–L7 and rows to sampled inputs; the color scale uses the original diagnostic's percentage KL convention. The plot establishes heterogeneity rather than an end-task accuracy bound.



(a) AWQ (b) SmoothQuant

Figure 4: Sensitivity to calibration-domain selection. A value of 1 denotes the best tested source for that evaluation domain; larger values indicate higher KL. "Common" denotes the commonsense domain. The strongest observed ratios are 3.16 for AWQ and 2.64 for SmoothQuant.

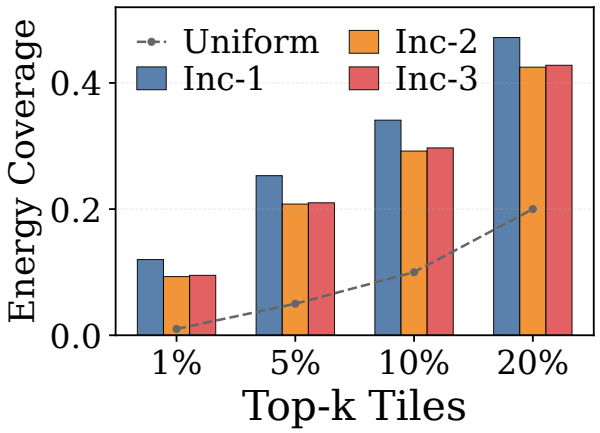


Figure 5: Concentration of increment energy across tiles. Energy-ranked selection covers a larger fraction of total increment energy than uniform selection at the same tile fraction.