BEYOND 4 BITS: DATA-CENTRIC POST-TRAINING QUANTIZATION AND RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization (PTQ) aims at compressing the weights of a large trained model using a limited compute and data budget. Down to 4 bits, state-of-the-art PTQ methods are near-lossless; below this limit, the tested scalar PTQ configurations collapse to near-random accuracy and model quality becomes acutely sensitive to choices such as the quantization representation and the calibration data. At such low bitwidths, recovery fine-tuning after quantization becomes crucial, and its effectiveness depends heavily on these same choices. Yet there is little systematic evidence on how the quantization method, the quality and task alignment of calibration data, and the design of the recovery procedure interact to determine the best final 2-bit model. We present a systematic study of 2-bit quantization and recovery fine-tuning across multiple model families, scales, and data distributions, and distill our findings into a practical, architecture-dependent set of best practices for efficiently obtaining an accurate 2-bit model. This recipe, which combines a vector codebook representation, a small task-aligned calibration mixture, and a short codebook-only distillation stage, restores 63–69% average accuracy across seven question-answering tasks at 2 bits (a 71–73% BF16 ceiling), versus the $\sim$29% random-accuracy floor of scalar PTQ (more so on MoE than dense models), at roughly half the compute budget of the strongest Quantization Aware Training (QAT) baseline. Our contribution is empirical: a systematic, architecture-aware map of the 2-bit design space, together with released code, Hessian mixtures, and checkpoints that make the findings directly actionable and reproducible.

1 INTRODUCTION

Quantization is a preferred route to serving Large Language Models (LLMs) under constrained memory and latency budgets (Frantar et al., 2023; Lin et al., 2024), and at 4 bits (W4A16) it is nearly free with state-of-the-art methods such as QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025). Quantization maps a continuous range of weight values onto a small signed discrete grid ($q \in \{-2, -1, 0, 1\}$ in the symmetric 2-bit case), with the goal of minimizing the error this discretization introduces (Gholami et al., 2022). Current post-training quantization (PTQ) methods typically compress a billion-parameter pretrained model with a few forward passes on a small calibration set on a single GPU (Frantar et al., 2023), shrinking its memory footprint by about $4\times$ at 4 bits (relative to BF16). This preserves accuracy to within a fraction of a point of the full-precision model, all without retraining (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024). Pushing to 3 or 2 bits would roughly halve the footprint again and is the natural next target. It is also where the standard scalar PTQ methods break (Egiazarian et al., 2024; Chee et al., 2023; Liu et al., 2024), as illustrated in Figure 1a.

Scalar post-training quantization (PTQ) exhibits a sharp performance cliff below 4 bits (Figure 1a). Methods that are nearly lossless at 4 bits, including QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025), degrade to near-random accuracy at 2 bits. This failure is particularly severe for reasoning models and on generative tasks, which additionally exhibit pronounced overthinking and repetitive looping at lower precisions (Lotfi et al., 2026). Existing approaches attempt to mitigate this collapse by trading additional computation for model quality. One line of work retains the PTQ framework but replaces scalar quantization with richer, codebook-based representations (Baalen et al., 2024; Liu et al., 2024; Egiazarian et al., 2024; Tseng et al., 2024a). Another replaces calibration-only PTQ with learning-based optimization, such as quantization-aware training (QAT)

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.

(Liu et al., 2023) or learned scalar grids (Dadgarnia et al., 2026). Learning-based methods, however, require substantially more data and computation (Liu et al., 2023; Du et al., 2024; Dadgarnia et al., 2026). Even after incurring these costs, 2-bit models continue to lag far behind their 4-bit counterparts, and no existing approach fully closes the gap.

We argue that a *trinity* of choices governs the quality of a 2-bit model, choices that practitioners usually treat as implementation details and mention only in passing: the quantization *representation*, the *calibration data*, and the *recovery fine-tuning procedure*. At 4 bits, these three choices hardly matter. Below 4 bits, they become decisive and are strongly coupled. Hence studying any *single choice* in isolation gives a misleading picture. As a result, no principled recipe exists for building the best possible 2-bit model; this work aims at bridging this gap.

We perform a systematic study of 2-bit PTQ and recovery methods across multiple model families and scales, and distill it into a concrete recipe. Moving from a scalar to a vector representation recovers most of the collapsed accuracy at a fixed compute budget. Aligning the calibration data with the target domain and curating it for quality, rather than simply scaling it up, recovers more. A lightweight recovery stage that updates only the codebook parameters with a distillation loss (Hinton et al., 2015) pushes performance even closer to 4-bit quality at a fraction of the compute of QAT and learning-based methods (Du et al., 2024; Dadgarnia et al., 2026) (Figure 1b). Together these choices yield a 2-bit model within 7.2 AvgQA points of the BF16 ceiling while using roughly half the training compute of the strongest QAT baseline. Our study spans five models across dense and MoE architectures: three primary models (Phi-4-mini, Llama-3.1-8B, Qwen3-MoE), with OLMoE and Qwen3-8B as auxiliary checks. We aim to deliver an empirically grounded, directly applicable recipe that gets the most out of existing quantizers. Our key contributions are:

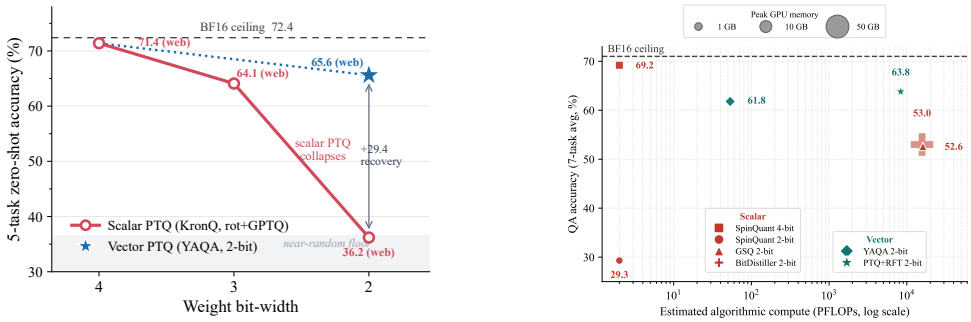
- **Representation drives the 2-bit cliff.** At a fixed compute budget, the tested scalar PTQ methods collapse to the $\sim 29\%$ random-accuracy floor in every family, while a vector codebook restores a usable model (average question-answering accuracy ≈ 63–69%).
- **Generation retention is strongly model-dependent.** At 2-bit vector quantization, QA retention is $\sim 90\%$ everywhere, but per-model generation retention (best 2-bit AvgGen, the GSM8K/HumanEval mean, vs BF16) spans 51% (Phi-4-mini) to 62% (Llama) to 101% (Qwen3-MoE). Hence, architecture and model scale, more than the quantizer, shape whether a 2-bit model can still reason and code.
- **Calibration and recovery are decisive.** A task-aligned, quality-filtered calibration mixture is crucial for generative tasks and a codebook-only distillation stage further narrows the generation gap to 4-bit at a fraction of QAT compute (Figure 1b).

Our study distills a simple recipe for accurate 2-bit models: quantize with a vector representation, calibrate on a small quality-filtered mixture matched to the target tasks, and recover with a short codebook-only distillation stage. It targets the limited compute and memory regime where a vector representation is deployable and only a small accuracy loss is acceptable, and its gains hold across model families and scales. We release our code and, upon acceptance, will also release the Hessian mixtures and checkpoints.

2 Related Work

Post-Training Scalar Quantization. Post-training quantization (PTQ) compresses a pretrained model from a small calibration corpus without retraining. **GPTQ** (Frantar et al., 2023) sets the near-lossless 4-bit weight-only baseline via approximate second-order error minimization; **AWQ** (Lin et al., 2024) and **SmoothQuant** (Xiao et al., 2023) rescale salient channels or migrate activation difficulty to weights, and rotation methods (**QuaRot** (Ashkboos et al., 2024), **SpinQuant** (Liu et al., 2025)) enforce weight incoherence before rounding. These scalar methods are near-lossless at 4 bits but collapse at 2 bits, where small uniform grids cannot represent LLM weight distributions. We take this collapse as our starting point.

Ultra-Low-Bit and Vector Quantization. Vector and codebook quantization jointly encode weights for better rate-distortion below 3 bits. **QuIP** (Chee et al., 2023) and **QuIP#** (Tseng et al., 2024a) pair incoherence preprocessing with lattice codebooks and fine-tuning; **AQLM** (Egiazarian et al., 2024), **GPTVQ** (Baalen et al., 2024), **VPTQ** (Liu et al., 2024), and **QTIP** (Tseng et al.,



(a) The 2-bit cliff.

(b) Trained baselines underperform vector PTQ.

Figure 1: **The 2-bit cliff and how we mitigate it.** **(a)** On Phi-4-mini (7-task QA average; BF16 71.0), scalar PTQ (SpinQuant) falls with bitwidth (69.2/50.1/29.3 at 4/3/2 bits, near the random floor), whereas a 2-bit vector representation (YAQA) stays far higher (61.8). **(b)** At 2 bits, every scalar method we evaluate trails the vector model (61.8), including trained GSQ (52.6) and BitDistiller (53.0); a short codebook-only recovery on the 2-bit YAQA model (PTQ+RFT) reaches 63.8 at roughly half their compute (8,349 vs 15,717–16,136 PFLOP). Marker area is proportional to peak GPU memory (PTQ $\sim$1, GSQ $\sim$2, recovery $\sim$9, full QAT $\sim$54 GB on Phi-4-mini). Per-method compute, memory, and accuracy are tabulated in Table 12.

2024b) push extreme compression through additive, hierarchical, or trellis-coded codebooks, and **YAQA** (Tseng et al., 2026) attains state-of-the-art 2-bit quality via a Fisher-weighted rounding objective. Others target 1-bit compression (**SqueezeLLM** (Kim et al., 2024), **BiLLM** (Huang et al., 2024), **PB-LLM** (Yuan et al., 2024), **OneBit** (Xu et al., 2024b)). These works study representation in isolation; we instead analyze how calibration data and recovery interact with the quantizer, narrowing the gap to BF16 at a fraction of QAT compute.

Learning-Based PTQ. Between calibration-only PTQ and full QAT, some methods *learn* the mapping to a quantization grid. **OmniQuant** (Shao et al., 2024) learns clipping thresholds block-wise, and **GSQ** (Dadgarnia et al., 2026) learns scalar-grid assignments through a Gumbel-Softmax relaxation. Because grid learning uses gradient signal but neither backpropagates through the full network nor runs multi-epoch training, we treat GSQ as an intermediate-cost baseline that isolates grid learning from representation choice.

Calibration Data. Most PTQ methods estimate statistics from 128–512 generic C4 or WikiText-2 samples, assuming content is interchangeable (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024; Egiazarian et al., 2024). Williams & Aletras (2024) first showed task-performance sensitivity, but only at 4 bits and mainly for web data. We extend this to 2 bits and show that task alignment and mixture composition, not sample count, dominate: small curated mixtures beat far larger generic corpora, and the gap widens as bitwidth drops.

Recovery for Quantized Models. Lightweight recovery narrows the post-quantization gap without full QAT, usually via knowledge distillation (Hinton et al., 2015): **LLM-QAT** (Liu et al., 2023), **BitDistiller** (Du et al., 2024), and **EfficientQAT** (Chen et al., 2025) update full weight matrices over multiple epochs and ignore codebook structure, while **LoRA/QLoRA** (Hu et al., 2022; Dettmers et al., 2023) reintroduce high-precision parameters that inflate the effective bitwidth. Our default recipe instead fine-tunes only the quantizer's native parameters (the vector codebook, or the scalar scale and zero-point), staying exactly at the target bitwidth and converging in a few steps (a LoRA variant is in Appendix A.15). We further expose a strong interaction between calibration composition and recovery, motivating a jointly designed calibration–recovery pipeline.

3 Background and Methodology

We build the PTQ-RFT pipeline, post-training quantization followed by recovery fine-tuning, from three ingredients. We first fix notation for the two quantization families we use, scalar and vector quantization, and identify for each the small set of parameters that recovery later updates (Sections 3.1–3.2). We then contrast the two ways to obtain a low-bit model, namely post-training quan-

tization (PTQ), which fits each layer locally to the full-precision model, and quantization-aware training (QAT), which reshapes the whole network end to end (Section 3.3). This contrast motivates our recovery stage (Section 3.4), a short fine-tuning step that stays *parameter-matched* to the quantized format and therefore does not raise the effective bitwidth.

3.1 Scalar Quantization

Scalar quantization maps each weight independently onto a shared one-dimensional grid of 2^b levels. In the general *asymmetric* form, round-to-nearest (RTN) over $[\min, \max]$ uses a scale s and integer zero-point z:

$$\begin{aligned} s &= \frac{\max - \min}{2^b - 1}, \qquad z = \operatorname{round}\left(-\frac{\min}{s}\right), \\ q &= \operatorname{clip}\left(\operatorname{round}\left(\frac{x}{s}\right) + z,\, 0,\, 2^b - 1\right), \qquad \hat{x} = s(q - z). \end{aligned} \tag{1}$$

The zero-point aligns the grid so that zero is represented exactly. Symmetric quantization instead fixes z and uses $s = \max/2^{b-1}$ with $\hat{x} = sq$, yielding a simpler offset-free kernel at the cost of one degree of freedom. Because RTN treats weights independently, its error grows rapidly at low bitwidths.

Error-aware PTQ. Methods such as GPTQ (Frantar et al., 2023) account for the effect of weight error on a layer's outputs. For $y = Wx$, calibration distribution X, and input Hessian $H_X = \mathbb{E}_{x\sim X}[xx^\top]$, they approximately solve

$$\min_{\hat{W}\in\mathcal{Q}} \mathbb{E}_{x\sim X}\big\|(W - \hat{W})x\big\|_2^2 = \min_{\hat{W}\in\mathcal{Q}} \operatorname{Tr}\big[(W - \hat{W})H_X(W - \hat{W})^\top\big], \tag{2}$$

where $\mathcal{Q}$ is the feasible set defined by the quantization format. GPTQ greedily quantizes coordinates while updating the remaining weights to compensate for accumulated error. This objective is a local second-order proxy for the full-model KL divergence; methods such as KronQ (Lee et al., 2026) improve the proxy using structured curvature.

The same formulation enables inexpensive calibration-mixture ablations. If H_j is estimated independently on domain j, then a mixture with proportions $\{\alpha_j\}$ has

$$H_{\boldsymbol{\alpha}} = \sum_j \alpha_j H_j, \qquad \alpha_j \geq 0, \qquad \sum_j \alpha_j = 1. \tag{3}$$

Thus, calibration composition can be varied by reweighting precomputed Hessians. Methods such as QAT and GSQ generally require a separate optimization run for each mixture.

Scalar methods differ primarily in how they determine the grid parameters. These are the parameters recovered by the recovery fine-tuning stage. It freezes the integer codes q at the target bitwidth and fine-tunes only s in the symmetric case or (s, z) in the asymmetric case, without introducing large high-precision weight updates.

3.2 Vector Quantization

Vector quantization changes the representation rather than only the precision of individual weights. It partitions a layer into d-dimensional sub-vectors and maps each to an entry in a learned codebook $\mathcal{C} = \{c_k\}_{k=1}^K \subset \mathbb{R}^d$. Reshaping W into vectors $\{w_i\}_{i=1}^N$ with assignments $a_i \in \{1, \ldots, K\}$ gives

$$\min_{\mathcal{C},\,\{a_i\}} \sum_{i=1}^N \|w_i - c_{a_i}\|_2^2, \qquad \hat{w}_i = c_{a_i}. \tag{4}$$

By representing points in $\mathbb{R}^d$ rather than values on a scalar grid, vector quantization exploits correlations among weights and remains effective at bitwidths where scalar formats degrade. Its effective bitwidth includes the per-vector indices and the amortized codebook storage.

Vector PTQ can use the same error-aware formulation as Equation 2, with $\mathcal{Q}$ replaced by the set of matrices representable by the chosen codebooks. VPTQ (Liu et al., 2024) uses channel-independent second-order optimization to solve this layerwise reconstruction problem efficiently. YAQA (Tseng

et al., 2026) instead derives its curvature from the full-model KL divergence and uses a Kronecker-factored approximation to guide adaptive rounding, including over trellis-coded quantization sets. The calibration-mixture construction in Equation 3 similarly applies to vector PTQ.

For vector formats, the recovery stage recovers the continuous codebook $\mathcal{C}$ while holding the discrete assignments $\{a_i\}$ fixed. Methods such as PV-tuning (Malinovskii et al., 2024) additionally optimize the assignments. Extending parameter-efficient recovery to discrete codebook or trellis assignments is a promising direction for future work.

3.3 Post-Training Quantization versus Quantization-Aware Training

PTQ and QAT optimize fundamentally different objectives. Layer-wise PTQ methods, including GPTQ (Frantar et al., 2023), KronQ (Lee et al., 2026), and YAQA (Tseng et al., 2026), minimize a local Hessian-weighted reconstruction error. This keeps each quantized layer close to its full-precision counterpart, leaving a small residual error that parameter-efficient fine-tuning can correct.

QAT instead optimizes the end-to-end objective through the quantizer, typically using the straight-through estimator (STE) (Liu et al., 2023), $\partial x_q/\partial x \approx 1$. Weights and intermediate representations may therefore depart from the full-precision model if doing so improves the task loss. This flexibility requires model-wide backpropagation and optimizer state, making QAT substantially more expensive than layer-wise PTQ.

Table 1: Persistent memory (GB) and estimated H100-80GB counts. PTQ and PTQ-RFT report 2-bit/4-bit configurations; H100 counts are ordered as Phi/Llama/Qwen.

Method	Phi-4 (GB)	Llama-8B (GB)	Qwen-MoE (GB)	H100s (P/L/Q)
PTQ	1.0/1.9	2.0/4.0	7.6/15.3	1/1/1
PTQ+RFT	9.0/10.0	18.8/20.8	71.5/79.1	1/1/1*
GSQ (2-bit)	2.2	5.0	17.0†	1/1/4†
Full QAT	53.8	112.4	427.0	1/2/6

Counts are persistent-state estimates and exclude activations, temporary buffers, and communication workspaces. *The 4-bit configuration occupies 79.1 GB and requires additional runtime capacity in practice. †Aggregate across four expert-sharded GPUs (4.3 GB per GPU).

Table 1 compares persistent memory for Phi-4-mini, Llama-3.1-8B, and Qwen3-30B-A3B, using $P = 3.84$B, 8.03B, and 30.5B parameters, respectively, and provides an approximate H100s count. An ideal b-bit PTQ checkpoint requires $bP/8$ bytes. PTQ followed by Recovery Fine-Tuning (RFT) requires approximately $2P + bP/8 + 12P/128$ bytes for the BF16 teacher, quantized model, and scale optimizer state. Full QAT requires approximately $14P$ bytes for the BF16 teacher, student weights and gradients, and two FP32 Adam moments; an optional FP32 master copy adds $4P$ bytes.

GSQ (Dadgarnia et al., 2026) lies between these regimes. It refines one transformer block at a time, avoiding a separate teacher and full-model optimizer state, but stores four BF16 assignment logits per optimized weight. Its logits, gradients, Lion momentum, and reconstructed weights require approximately $26N$ bytes for a block with N optimized weights. This yields 2.2 GB for Phi-4-mini, 5.0 GB for Llama-3.1-8B, and 17.0 GB for Qwen3-30B-A3B across four expert-sharded GPUs.

Overall, GSQ optimizes a discrete assignment for each weight in the active block, whereas PTQ+RFT freezes weights and assignments and updates only continuous quantizer parameters. PTQ+RFT therefore pays for a full-precision teacher but retains a much smaller trainable state.

3.4 Recovery Fine-Tuning

Recovery fine-tuning (RFT) restores accuracy lost to low-bit quantization without the cost of full QAT. In its base form, it freezes the quantized backbone and updates only the parameters already present in the quantized representation: the scales s and zero-points z for scalar formats and the codebook $\mathcal{C}$ and assignments $\mathcal{A}$ for vector formats. Since assignments are discrete and difficult to optimize, we only consider the codebook $\mathcal{C}$ for recovery fine-tuning. This preserves the target storage format and effective bitwidth while applying uniformly to scalar and vector quantizers.

Base objective. We train against the full-precision model as teacher, using logit-level distillation. Let z_t and z_s denote the teacher and student logits, respectively. Our objective combines a temperature-scaled forward KL divergence with next-token cross-entropy:

$$\mathcal{L} = \alpha\,\mathcal{L}_{\mathrm{CE}} + (1-\alpha)\,\mathcal{L}_{\mathrm{KD}}, \qquad \mathcal{L}_{\mathrm{KD}} = \tau^2\,\mathrm{KL}\big(\sigma\big(\tfrac{z_t}{\tau}\big)\,\big\|\,\sigma\big(\tfrac{z_s}{\tau}\big)\big)\,, \tag{5}$$

where $\sigma(\cdot)$ is the softmax, τ is the distillation temperature, and $\alpha \in [0, 1]$ balances the two terms. This objective uses only model logits and introduces no additional trainable parameters. We use it for all base experiments; alternative divergences, temperatures, and top-k logit truncation appear in Appendix A.8.

Adapter-based recovery. Recovery can also add parameter-efficient modules such as LoRA adapters (Hu et al., 2022). Prior work recovers frozen quantized models with low-rank adapters (QLoRA (Dettmers et al., 2023), QA-LoRA (Xu et al., 2024a), LoftQ (Li et al., 2024)); these are complementary to quantizer-only recovery. Adapters, however, add deployment overhead and may remain in half precision unless merged and requantized, so we use quantizer-only recovery as our default format-preserving setting. Appendix A.15 evaluates joint quantizer–adapter recovery.

4 Experiments and Results

We organize the evaluation around four factors that determine the viability of weight-only 2-bit quantization: representation, optimization budget, calibration data, and recovery parameterization. Specifically, we ask (i) where scalar PTQ fails as precision drops from 4 to 2 bits; (ii) whether vector codebooks avert this failure across dense and mixture-of-experts (MoE) architectures; (iii) how learnable scalar PTQ and QAT compare with vector PTQ; and (iv) how calibration and codebook recovery trade off language understanding, mathematical reasoning, and code generation.

4.1 Experimental setup

Models and quantization. We evaluate Phi-4-mini-instruct (Microsoft et al., 2025) (3.8B, abbreviated Phi-4-mini), Llama-3.1-8B (Grattafiori et al., 2024), and Qwen3-30B-A3B (Yang et al., 2025), a 30B-parameter MoE with approximately 3B active parameters. The no-recovery scalar baselines are SpinQuant (Liu et al., 2025) and QuaRot (Ashkboos et al., 2024). At 2 bits, scalar weights use group size 128 with a half-precision per-group scale, or approximately 2.125 bits per quantized weight (2.12 in the appendix tables; an asymmetric zero-point variant would be 2.25). The vector baselines are VPTQ (Liu et al., 2024), which uses a $65{,}536$-entry codebook over 8-dimensional vectors (2.00 bits/weight), and YAQA (Tseng et al., 2026), a Fisher-weighted trellis quantizer (approximately 2.26 bits/weight). We also compare AQLM (Egiazarian et al., 2024) as a named 2-bit vector baseline, GSQ (Dadgarnia et al., 2026) as a group-128 scalar PTQ method that optimizes discrete assignments layer by layer, and BitDistiller (Du et al., 2024) as group-128 scalar QAT with self-distillation. Thus, GSQ and BitDistiller receive additional optimization unavailable to the headline no-recovery PTQ methods; their comparisons test whether scalar optimization, rather than representation, is sufficient at 2 bits. For dense models we quantize attention and MLP projections. For Qwen3-MoE we quantize attention and every expert FFN; the headline experiments retain router gates, embeddings, normalization parameters, and the language-model head in half precision. Appendix A.14 separately quantizes the routers. The scalar rate includes per-group metadata. The vector rates are reported index rates and exclude amortized codebook storage and retained fp16 tensors; YAQA consequently spends about 13% more index bits than VPTQ. The conclusion also holds against trained scalar baselines. On Llama, the best task-complete 2-bit BitDistiller checkpoint reaches 60.3 AvgQA and GSQ reaches 57.0, versus 65.7 for YAQA. On Qwen3-MoE, the best task-complete 2-bit BitDistiller checkpoint reaches 55.9 AvgQA, versus 68.7 for VPTQ. BitDistiller uses full-model QAT with self-distillation, whereas GSQ optimizes discrete scalar assignments after GPTQ (Frantar et al., 2023) initialization; neither is a no-training PTQ comparison (Appendix A.11). We specify every method in full in the appendix, including per-method quantization and recovery configurations (Table 7, Appendix A.2), the evaluation protocol (Appendix A.1), and training-compute, peak-memory, and deployment profiles (Table 11, Appendices A.5–A.6).

Calibration and evaluation. We compare web, instruction, reasoning, code, and convex mixtures of these calibration sources. MMLU (Hendrycks et al., 2021a) and GSM8K (Cobbe et al., 2021) use 5-shot prompting. ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and PIQA (Bisk et al., 2020) use length-normalized accuracy; WinoGrande (Sakaguchi et al., 2021) and LAMBADA (Paperno et al., 2016) use accuracy; HumanEval (Chen et al., 2021) uses pass@1; IFEval (Zhou et al., 2023) uses prompt-strict accuracy. We report AvgQA, the mean of MMLU, WinoGrande, ARC-Easy, ARC-Challenge, HellaSwag, LAMBADA, and PIQA, and AvgGen, the mean of GSM8K and HumanEval.

Table 2: **Quantization ladder.** Best QA and generation at each precision under *web calibration*, selected separately for QA and generation, *without recovery*. Scalar entries report the better of SpinQuant (^{S}) and QuaRot (^{Q}) (on Qwen3-MoE, QuaRot uses expert-wise rotation); the 2-bit vector (VQ) entries report the better of YAQA (^{Y}) and VPTQ (^{V}). All entries use web calibration; superscripts mark the winning method.

	BF16		Scalar 4-bit		Scalar 2-bit		Vector 2-bit	
Model	QA	Gen	QA	Gen	QA	Gen	QA	Gen
Phi-4-mini	71.0	73.2	69.3^{Q}	65.8^{S}	29.3^{S}	0.0	63.4^{Y}	12.2^{Y}
Llama-3.1-8B	73.1	42.9	71.8^{S}	38.0^{Q}	29.0^{Q}	0.0	65.1^{Y}	9.6^{Y}
Qwen3-30B-A3B	72.6	61.8	70.3^{S}	66.2^{S}	29.4^{S}	0.0	67.7^{V}	54.0^{Y}

Table 3: **Cross-paradigm comparison at 2 bits.** Best 2-bit AvgQA (a) and AvgGen (b) per model. VPTQ and YAQA select the best calibration recipe for each metric independently (recipes in §A.10); AQLM uses one fixed reason+web recipe. GSQ is scalar PTQ with layer-wise discrete optimization; BitDistiller is full-model QAT with self-distillation; AQLM, VPTQ, and YAQA are vector PTQ. Bold marks the best per model.

(a) QA (AvgQA ↑).

		Trained scalar		Vector PTQ		
Model	BF16	GSQ	BitDist.	AQLM	VPTQ	YAQA
Phi-4-mini	71.0	52.6	53.0	57.3	53.9	**63.3**
Llama-3.1-8B	73.1	57.0	60.3	60.5	59.8	**65.7**
Qwen3-MoE	72.6	29.4	55.9	**71.2**	68.7	66.7
OLMoE-1B-7B	68.3	29.6	49.1	**65.5**	65.2	64.0

(b) Generation (AvgGen ↑).

		Trained scalar		Vector PTQ		
Model	BF16	GSQ	BitDist.	AQLM	VPTQ	YAQA
Phi-4-mini	73.2	21.6	18.4	27.0	11.1	**37.4**
Llama-3.1-8B	42.9	20.0	16.2	17.9	16.1	**26.6**
Qwen3-MoE	61.8	0.0	45.7	61.2	58.9	**62.4**
OLMoE-1B-7B	32.4	0.0	0.9	**21.6**	21.4	20.6

Recovery. Recovery freezes the quantized backbone and updates only the separate vector codebook. The reference run uses a BF16 teacher, forward KL, temperature $\tau = 1$, full-vocabulary logits, and web recovery data. We vary one factor at a time. This protocol tests codebook-restricted post-quantization recovery rather than full QAT.

4.2 Representation determines whether 2-bit PTQ works

Scalar PTQ has a sharp 2-bit cliff. Table 2 compresses the full per-task results into QA and generation means. At 4 bits, the best scalar method stays within 1.3–2.3 QA points of BF16. At 2 bits, every evaluated scalar method collapses to approximately 29% QA and zero generation, with the per-model generation curves in Appendix A.21. We evaluated both methods on all models, applying QuaRot to Qwen3-MoE via expert-wise rotation. The failure is therefore not specific to one rotation method, dense architecture, or model scale. Named 2-bit scalar methods show the same pattern (Appendix A.9). Generation reveals a distinct MoE regime already at 4 bits: Phi-4-mini and Llama lose 7.4 and 4.9 AvgGen points, whereas Qwen3-MoE gains 4.4 points (66.2 versus 61.8). We treat the latter as full retention at the resolution of this evaluation, not as evidence that quantization improves generation.

Vector codebooks revive 2-bit models. Changing only the representation raises 2-bit QA from the random floor to 63.4 with YAQA on Phi-4-mini, 65.1 with YAQA on Llama, and 67.7 with VPTQ on Qwen3-MoE under web calibration. The effect on generation is architecture-dependent: dense models retain little of their BF16 generation score, whereas Qwen3-MoE reaches 54.0 AvgGen with YAQA. Thus, bitwidth alone is insufficient to predict failure; representation, architecture and scale jointly determine the 2-bit regime.

4.3 Vector PTQ outperforms trained scalar quantization

Table 3 separates representation from optimization budget. On Llama, YAQA exceeds GSQ by 8.7 AvgQA points and BitDistiller by 5.4 points, and even fixed-recipe AQLM edges BitDistiller by 0.2 points. On Qwen3-MoE, fixed-recipe AQLM exceeds BitDistiller by 15.3 points; on OLMoE, it edges the best-recipe VPTQ result by 0.3 points. The OLMoE and Qwen3-MoE GSQ checkpoints both collapse to the random floor (29.6 and 29.4 AvgQA, zero AvgGen). Vector PTQ therefore remains stronger even though BitDistiller uses full-model QAT and GSQ explicitly optimizes scalar assignments, and a QTIP trellis-coded control reproduces this vector advantage (Table 22, Appendix A.13). These are checkpoint comparisons rather than compute-matched comparisons, but

they show that additional scalar optimization alone does not recover the capability retained by vector representations. Appendix A.4 reports the gaps explicitly, and Appendix A.11 gives the complete trained-baseline task vectors and calibration sweeps.

Within vector PTQ, the preferred method still depends on architecture. YAQA improves the best AvgQA score over VPTQ by 9.1 points on Phi-4-mini and 5.9 points on Llama. On Qwen3-MoE, VPTQ is instead 2.0 points better while using fewer index bits. A third dense model, Qwen3-8B, reproduces this dense ordering (YAQA 63.7 versus VPTQ 60.8 AvgQA under reason+web; Appendix A.17). The per-domain results in Appendix A.3 reproduce this ordering across calibration sources, ruling out selection of a single favorable recipe. Against AQLM, the best YAQA result improves AvgQA from 57.3 to 63.0 on Phi-4-mini and from 60.5 to 65.7 on Llama (Appendix A.12). On the sparse models, fixed-recipe AQLM instead gives the strongest AvgQA: 71.2 on Qwen3-MoE and 65.5 on OLMoE. Because AQLM uses one fixed 50/50 reasoning/web calibration while these are best-over-recipe numbers, the matched YAQA comparison is 62.6 versus 57.3 on Phi-4-mini and 65.7 versus 60.5 on Llama. YAQA's dense-model gains also use about 13% more index bits than VPTQ, but a bit-matched control confirms this advantage persists at equal rate (Table 8). OLMoE provides an independent sparse-model check: its best 2-bit VPTQ result retains 65.2/68.3, or 95.5%, of the BF16 AvgQA score, while AQLM retains 65.5/68.3, or 95.9%. Together with Qwen3-MoE AQLM's 71.2/72.6 (98.1%) retention, this suggests that strong 2-bit vector-quantized retention extends beyond a single MoE family. It does not, however, identify the architectural mechanism.

4.4 Calibration chiefly controls generation

Calibration composition remains important after the representation is fixed. Table 4 retains representative recipes with complete task coverage across all three models. QA varies by 7.5 points on Phi-4-mini, 4.9 on Llama, and 3.9 on Qwen3-MoE. Generation is substantially less stable, varying by 26.7, 17.2, and 13.5 points, respectively. Reasoning calibration maximizes Llama generation, while broad mixtures remain competitive for QA. On Qwen3-MoE, the targeted reasoning/web/instruction mixture RWI442 gives the best generation score, while a balanced reasoning/web mixture gives the best QA. The complete cross-model sweep appears in Appendix A.7.

Table 4: **Calibration composition primarily changes generation.** QA and generation means for representative 2-bit YAQA recipes. Bold marks the best result over the full sweep; Δ_{full} uses every available recipe, including model-specific mixtures omitted from this compact table.

	Phi-4-mini		Llama-3.1-8B		Qwen3-MoE	
Recipe	QA	Gen	QA	Gen	QA	Gen
Web	**63.4**	12.2	65.1	9.6	66.0	54.0
Reasoning	58.5	**37.4**	62.1	**26.6**	64.5	61.8
Code	55.9	18.7	60.8	22.6	62.8	48.9
Reason+web (0.5/0.5)	62.6	24.7	**65.7**	17.8	**66.7**	56.5
Webreason (0.7/0.3)	63.0	27.7	65.5	15.1	65.8	56.2
All-4 (0.25×4)	58.8	19.9	65.2	20.1	66.0	59.3
RWI442 (0.4/0.4/0.2)	59.5	17.1	62.7	12.2	66.5	**62.4**
Δ_{full}	7.5	26.7	4.9	17.2	3.9	13.5

Generation retention separates dense models from the MoE. Selecting methods separately for each metric, the strongest task-complete QA/Gen results are 63.4/37.4 on Phi-4-mini, 65.7/26.6 on Llama, and 68.7/62.4 on Qwen3-MoE. These correspond to 89%/51%, 90%/62%, and 95%/101% of the respective BF16 scores (AvgQA/AvgGen for this task-complete selection), matching the 51–101% per-model *generation* retention band quoted in the introduction and Table 6. QA retention is therefore similar across families, whereas generation retention spans more than a factor of two, as the per-task retention heatmaps and QA–generation scatterplots make explicit (Appendix A.20, Figures 3–6). This result, together with the reversed VPTQ–YAQA ranking, makes architecture a first-order variable in 2-bit quantization. We note, however, that a second MoE (OLMoE) retains far less generation than Qwen3-MoE ($\approx 66\%$ vs $\approx 102\%$ of BF16), so architecture strongly influences, but does not by itself determine, generation retention.

4.5 Codebook-only recovery helps

Recovery improves capability without restoring full-precision parity (Table 5), and does so at a fraction of QAT compute, placing recovered checkpoints on the accuracy–compute Pareto front (Appendix A.22). On Phi-4-mini, codebook-only distillation raises AvgQA from 61.8 to 64.1; reasoning data raises GSM8K from 44.9 to 62.5 and HumanEval from 8.5 to 36.6. Recovery data matters more than KD details: loss direction, teacher precision, temperature, and top-k truncation

Table 5: **Best-recipe codebook recovery (full suite).** Best recovery recipe (forward KL, τ=1, full-vocabulary logits, BF16 teacher) applied to the single 2-bit YAQA no-recovery base shown for each model, so every recovery row shares that common base and its change isolates the recovery-data effect; the row names the recovery-data composition. AvgQA is the seven-task mean; GSM8K, HumanEval (pass@1), and IFEval (prompt-strict) are separate. Bold marks the best recovered value per model; codebook-only recovery yields at most small IFEval gains (e.g., Phi $37.2 \rightarrow 39.2$), and Llama's base-model IFEval is intrinsically low (Appendix A.19).

Model	Recovery data	AvgQA	GSM8K	HEval	IFEval
Phi-4-mini	BF16 ceiling	71.0	81.9	64.6	56.6
	2-bit YAQA (no recovery)	61.8	44.9	8.5	37.2
	web	63.8	40.5	19.5	27.4
	reasoning	61.0	**62.5**	**36.6**	27.7
	reason_web	63.7	51.6	26.8	**36.0**
	webreason	**64.1**	52.5	23.8	31.6
	rw64 (recommended)	63.8	57.7	28.7	34.2
Llama-3.1-8B	BF16 ceiling	73.1	50.9	34.8	8.9
	2-bit YAQA (no recovery)	65.7	22.1	13.4	7.4
	webreason	64.8	17.7	17.7	6.8
	all-4	64.6	17.1	**23.2**	8.7
Qwen3-30B-A3B	BF16 ceiling	72.6	90.0	33.5	28.3
	2-bit YAQA (no recovery)	66.0	73.7	33.5	24.4
	web (recovery)	**67.8**	**76.0**	**45.1**	22.0

each move AvgQA by at most 0.2, and a 4-bit teacher matches BF16 (63.6 vs. 63.5). The trend transfers: Llama gains 9.8 HumanEval points for 1.1 AvgQA, while Qwen3-MoE improves both AvgQA ($66.0 \rightarrow 67.8$) and HumanEval ($33.5 \rightarrow 45.1$); a dedicated Qwen recovery fine-tuning sweep confirms that broad web and mixture data outperform narrow instruction, reasoning, and code recovery (Table 26, Appendix A.16). IFEval remains a limitation (Appendices A.8 and A.19; optional LoRA in Appendix A.15).

MoE robustness. Quantizing Qwen3-MoE's router to 2 bits changes accuracy by at most 0.2, while OLMoE retains 64.7/68.3 AvgQA at 2-bit VPTQ. Thus fp16 routing does not explain the 5–6 point MoE retention advantage; its mechanism remains open (Appendices A.14, A.18, and A.24).

5 A Practical Recipe for 2-Bit Models

The decision order is: **representation** first (vector, not scalar), then **architecture** (YAQA for dense models; VPTQ for MoEs), then **data** (task-aligned Hessian calibration and short codebook-only recovery with a disposable 4-bit teacher), as summarized in Table 6. Calibration composition moves generation far more than QA, and the KD hyperparameters barely matter once we choose the recovery data. Keep the MoE router in fp16 as a cheap default, and report QA and generation separately, since generation retention spans a much wider band (51–101%) than QA. This single-GPU recipe yields 63–69% AvgQA at 2 bits across the models we study.

Table 6: **2-bit decision guide.** Recommended choice and evidence per decision (Section 4); AvgQA unless noted.

Decision	Recommendation	Evidence
Representation	Vector (not scalar)	$29 \rightarrow 63$–69
Dense quantizer	YAQA	$+9.1/+5.9$ vs. VPTQ
MoE quantizer	VPTQ	$+2.0$ vs YAQA; AQLM best QA
Calibration	Hessian mixture	moves gen. $\gg$ QA
Recovery	Codebook-only KD	$61.8 \rightarrow 64.1$
KD knobs	Defaults (τ=1)	≤ 0.2 AvgQA
Teacher	4-bit (not BF16)	$63.6 \approx 63.5$
Router	keep fp16	< 0.2 (not cause)
Decoding	Rep. penalty 1.3	$0.58 \rightarrow 0.70$ MATH
Gen. boost	+LoRA (optional)	HEval $\rightarrow 32.9$
Reporting	split QA/gen	ret. 51–101%

Decoding matters at 2 bits. A final lever acts at inference. On reasoning models with an explicit thinking mode, 2-bit quantization amplifies an *overthinking* pathology: on MATH-500, quantizing Qwen3-MoE to 2 bits doubles the hesitation-marker rate ($15.5 \rightarrow 32.2$ per 1k tokens) and raises the 4-gram loop score ($0.39 \rightarrow 0.58$), lowering accuracy from 0.675 to 0.50. This is a decoding pathology, not lost capability: a mild repetition penalty of 1.3 restores 2-bit accuracy to the BF16 level ($0.575 \rightarrow 0.70$) while halving the loop score (Tables 32 and 33, Appendix A.23); a small repetition penalty is thus a zero-cost default for 2-bit reasoning models.

6 Conclusion

The 2-bit cliff is primarily a representation failure: scalar methods collapse, whereas vector codebooks recover strong QA and, with data-centric calibration and recovery, substantial generation. Instruction following and the mechanism behind MoE robustness remain interesting directions for future work.

USE OF AI TOOLS

All ideas presented in this paper are original to the authors or build upon the related work cited herein. AI-based tools were used only in an assistive capacity: to sharpen the clarity of the authors' initial drafts, to help typeset the paper in LaTeX, to rephrase sections, to improve the presentation of figures and tables, to correct grammatical errors and make sentences more salient, and to act as a coding assistant that implements the authors' ideas. All technical content, claims, and conclusions were conceived, verified, and approved by the authors.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dfqsW38v1X.

Mart Van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Artem Bolshakov, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. GPTVQ: The blessing of dimensionality for LLM quantization. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024. URL https://openreview.net/forum?id=sFWttLybEu.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

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

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

Alireza Dadgarnia, Soroush Tabesh, Mahdi Nikdan, Michael Helcig, Eldar Kurtic, Maximilian Kleinegger, and Dan Alistarh. GSQ: Highly-accurate low-precision scalar quantization for LLMs via gumbel-softmax sampling. *arXiv preprint arXiv:2604.18556*, 2026.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

DaYou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 102–116, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.7. URL https://aclanthology.org/2024.acl-long.7/.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=5mCaITRTmO.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, et al. A framework for few-shot language model evaluation. https://github.com/EleutherAI/lm-evaluation-harness, 2023.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and XIAOJUAN QI. BiLLM: Pushing the limit of post-training quantization for LLMs. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=qOl2WWOqFg.

Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=0jpbpFia8m.

Donghyun Lee, Yuhang Li, Ruokai Yin, and Priyadarshini Panda. Kronq: LLM quantization via kronecker-factored hessian. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=A3YwGs0WWT.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LzPWWPAdY4.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. VPTQ: Extreme low-bit vector post-training quantization for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8181–8196, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.467. URL https://aclanthology.org/2024.emnlp-main.467/.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Sanae Lotfi, Polina Kirichenko, Steven Li, and Zechun Liu. Quantized reasoning models think they need to think longer, but they do not. *arXiv preprint arXiv:2606.00206*, 2026.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Pavlovich Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtárik. PV-tuning: Beyond straight-through estimation for extreme LLM compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YvA8UF0I37.

Microsoft, Abdelrahman Abouelenin, Atabak Ashfaq, et al. Phi-4-Mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs, 2025.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xXTkbTBmqq.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP$\#$: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=9BrydUVcoe.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=7sdkLVuYCU.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. In *International Conference on Machine Learning (ICML)*, 2026. YAQA: Yet Another Quantization Algorithm.

Miles Williams and Nikolaos Aletras. On the impact of calibration data in post-training quantization and pruning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10100–10118, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.544. URL https://aclanthology.org/2024.acl-long.544/.

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 (ICML)*, 2023.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=WvFoJccpo8.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. Onebit: Towards extremely low-bit large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=ZwiG9KjfHV.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report, 2025.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. PB-LLM: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BifeBRhikU.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Additional Experimental Details and Results

A.1 Quantization and evaluation details

Quantized modules. For Phi-4-mini and Llama-3.1-8B, quantized modules include attention and all MLP linear projections. The language-model head, token embeddings, and normalization parameters remain in half precision. For Qwen3-30B-A3B, we quantize attention and every expert FFN; the headline experiments retain router gates, embeddings, normalization parameters, and the language-model head in half precision. The router ablation in Appendix A.14 relaxes this choice.

Effective precision. Scalar 2-bit experiments use group size 128 with a half-precision per-group scale, giving approximately 2.125 bits per quantized weight (rounded to 2.12 in Tables 7 and 13); an asymmetric variant that adds a half-precision per-group zero-point would instead cost 2.25 bits. All reported scalar runs use the scale-only (2.12) format. VPTQ (Liu et al., 2024) uses a $(65{,}536, 8)$ codebook and reports a 2.00-bit index rate. YAQA (Tseng et al., 2026) uses a Fisher-weighted trellis and reports approximately 2.26 index bits per weight. The vector rates exclude amortized codebook storage and retained fp16 tensors; YAQA therefore spends about 13% more index bits than VPTQ. Both methods retain a separate codebook suitable for recovery.

Metrics. MMLU and GSM8K use 5-shot evaluation. ARC-Easy, ARC-Challenge, HellaSwag, and PIQA use length-normalized accuracy. HumanEval reports pass@1 and IFEval reports prompt-strict accuracy. AvgQA averages MMLU, WinoGrande, ARC-Easy, ARC-Challenge, HellaSwag, LAMBADA, and PIQA; AvgGen averages GSM8K and HumanEval.

Table 7: **Per-method configuration summary.** Paradigm, effective bits/weight (*Eff.*, cf. Table 13), the parameters each method trains, and the calibration or recovery data. All methods use group size 128 and the shared evaluation protocol (§A.1).

Method	Paradigm	Eff. b/w	Trained parameters	Calibration / recovery data
QuaRot	scalar PTQ (rotation)	2.12	none (RTN)	128 sequences
SpinQuant	scalar PTQ (rotation)	2.12	rotation matrices	128 sequences
GSQ	learned scalar	2.12	grid assignments	0.1B-token, 10 epochs
BitDistiller	full QAT	2.12	all weights	1-epoch mixture
VPTQ	vector PTQ	2.00	codebook (post-hoc)	per-domain Hessians
YAQA	vector PTQ	2.26	codebook (post-hoc)	per-domain Hessians
AQLM	vector (learned)	2.00	codebooks + fine-tune	128 seq., reason+web
PTQ+RFT	codebook recovery	2.26	codebook only	task-aligned mixtures

Evaluation and reproducibility. We score all tasks through an `lm-evaluation-harness`-based split-task orchestrator (Gao et al., 2023). Multiple-choice tasks (ARC-Easy, ARC-Challenge, HellaSwag, PIQA, WinoGrande, LAMBADA) use log-likelihood scoring; MMLU and GSM8K use 5-shot prompting; GSM8K and HumanEval are decoded greedily (temperature 0, no sampling) and scored by strict-match and pass@1, respectively. Calibration and recovery sample counts are the audited per-method values used in the compute accounting (§A.5): 128 sequences for scalar calibration and 128–2,048 sequences per domain for the vector Hessians. Training and recovery runs use a fixed random seed (42).

A.2 Per-method quantization configurations

This section consolidates the configuration of every quantization and recovery method we evaluate. Table 7 summarizes the key settings; the paragraphs below give the details. Unless noted, all methods target 2-bit weights at group size 128, share the calibration and evaluation protocol of §A.1, and use a fixed random seed (42).

Scalar PTQ: QuaRot and SpinQuant. Both apply an incoherence rotation before round-to-nearest quantization at group size 128 with half-precision per-group scale and zero-point. QuaRot uses a fixed randomized Hadamard transform with GPTQ-style error feedback, whereas SpinQuant learns the rotation on a small calibration set; on Qwen3-MoE we apply the rotation expert-wise. Both consume 128 calibration sequences and add no trained weights, giving our no-recovery scalar reference.

Vector PTQ: VPTQ. VPTQ maps every 8-dimensional weight sub-vector to one of 65,536 (2^{16}) codebook entries (2.00 index bits/weight) through channel-independent second-order (input-Hessian) optimization over a shared Hadamard frame. We collect input Hessians per calibration domain. The bit-matched control of §A.3 instead uses a 262,144 (2^{18}) codebook (2.25 bits/weight).

Vector PTQ: YAQA. YAQA quantizes onto a trellis codebook using Fisher-weighted, Kronecker-factored curvature derived from the full-model KL objective, at $\sim$2.26 index bits/weight, and retains a separate, fine-tuning-ready codebook under the same incoherence rotation. We estimate curvature from 128–2,048 sequences per calibration domain.

AQLM. AQLM (additive quantization) uses a single codebook with 16-bit indices over 8-dimensional groups (2.00 bits/weight). Each layer's codebook is initialized by K-means ($\leq$ 100 iterations) and refined by beam search (beam size 1, up to 1000 epochs of 100 steps, relative-MSE tolerance 10^{-2}, learning rate 10^{-4}), followed by a global end-to-end fine-tune ($\leq$ 10 epochs with early stopping after 3, learning rate 10^{-5}, batch 32). Calibration is 128 sequences of length 2048 from a $50/50$ reasoning/web mixture (§A.12).

GSQ (learned scalar). GSQ learns discrete grid assignments through a Gumbel-softmax relaxation, initialized with GPTQ (512 sequences). It optimizes one transformer block at a time with detached inputs for 10 epochs over a 0.1B-token calibration set (48,828 sequences of length 2048) at group size 128, using a Lion optimizer. Block-wise processing keeps its peak memory low despite a high total FLOP count (Tables 12 and 13).

Table 8: **Bit-matched vector-quantizer control (Phi-4-mini).** Raising VPTQ from 2.00 to 2.25 bits/weight, matching YAQA's 2.26, closes about 78% of the web QA gap (51.5 → 60.7 versus YAQA 63.3), but YAQA still leads at equal bits, so the trellis *representation*, not bit count alone, explains the residual gap (curvature and optimization also differ). AvgQA is the seven-task mean; AvgGen is the mean of GSM8K and HumanEval (pass@1). YAQA leads AvgQA on both recipes (+2.6 web, +4.9 reasoning) and reasoning-recipe generation, while web-recipe generation is low for both. Bold marks the best per column; dashes are cells we did not log (the 2.00-bit VPTQ generation was not measured per domain).

	web		reasoning	
Method (bits/weight)	AvgQA	AvgGen	AvgQA	AvgGen
VPTQ (2.00)	51.5	–	41.0	–
VPTQ (2.25)	60.7	**15.4**	53.6	24.3
YAQA (2.26)	**63.3**	12.2	**58.5**	**37.4**

Table 9: **AvgQA by vector quantizer and calibration domain.** Bold marks the better method within each model/domain.

Model	Method	Web	Instruct	Reason	Code	Mixture
Phi-4-mini	VPTQ	51.5	53.9	41.0	44.4	47.8
	YAQA	**63.3**	**56.8**	**58.5**	**55.9**	**58.8**
Llama-3.1-8B	VPTQ	57.6	59.8	45.7	53.4	54.5
	YAQA	**65.1**	**63.9**	**62.1**	**60.8**	**64.3**
Qwen3-30B-A3B	VPTQ	**67.7**	**68.7**	**64.7**	**67.4**	**67.1**
	YAQA	66.0	64.9	64.5	62.8	66.0

BitDistiller (full QAT). BitDistiller performs full-model quantization-aware training with asymmetric activation clipping and confidence-aware self-distillation (CAKLD, $\beta \approx 0.84$) on top of a Hadamard-rotated 2-bit model at group size 128. We train for one epoch with FSDP full-shard across 8 GPUs, sequence length 1024, per-device batch 1 and gradient accumulation 8 (effective batch 64), learning rate 2×10^{-5} with cosine decay and 10% warmup. The recovery mixture holds ∼500k sequences for the dense models and 47,616 for Qwen3-MoE (the step-800 checkpoint; §A.11).

Codebook-only recovery (PTQ+RFT). Our recovery stage freezes the quantized backbone and the discrete assignments and updates only the continuous codebook (or the scale and zero-point for scalar formats), leaving the effective bitwidth unchanged. It distills from a BF16 (or disposable 4-bit) teacher with a temperature-1 forward-KL term plus next-token cross-entropy over the full vocabulary, using per-device batch 1, gradient accumulation 8, learning rate 10^{-4}, a cosine schedule with 500 warmup steps, and sequence length 1024. Recovery data are the task-aligned mixtures of §A.8; the optional LoRA variant adds rank-16 (α=32) adapters (§A.15).

A.3 Per-domain vector-quantizer comparison

Table 9 exposes the architecture-dependent VPTQ–YAQA reversal at each base calibration domain. YAQA wins every completed dense-model comparison. VPTQ wins all five Qwen3-MoE domains (reasoning is within 0.2 points). Phi-4-mini YAQA reasoning and code completed after the older aggregate table and reach AvgQA scores of 58.5 and 55.9, respectively.

YAQA's advantage survives bit-matching. Because YAQA spends ∼13% more index bits than VPTQ (2.26 versus 2.00 bits/weight), we ran a bit-matched control on Phi-4-mini: VPTQ with a 262,144-entry codebook (2.25 bits/weight, matching YAQA). Raising VPTQ to the same budget lifts web-calibrated AvgQA from 51.5 (at 2.00 bits) to 60.7, but YAQA still leads at equal bits: 63.3 versus 60.7 AvgQA on web (+2.6) and 58.5 versus 53.6 on reasoning (+4.9). Raising the index rate to 2.25 bits thus closes about 78% of the original gap (9.2 of 11.8 AvgQA points); the residual 2.6-point YAQA advantage reflects its trellis representation and differing curvature (Table 22), not the bit budget alone (Table 8).

Table 10: **Vector PTQ versus trained scalar quantization at 2 bits.** AvgQA (↑). "Best VQ" selects the strongest task-complete VPTQ or YAQA checkpoint for each model. Δ is Best VQ minus the scalar baseline.

Model	Scalar baseline	AvgQA	Best VQ	AvgQA	Δ
Phi-4-mini	GSQ	52.6	YAQA	63.0	+10.4
Phi-4-mini	BitDistiller	53.0	YAQA	63.0	+10.0
Llama-3.1-8B	GSQ	57.0	YAQA	65.7	+8.7
Llama-3.1-8B	BitDistiller	60.3	YAQA	65.7	+5.4
Qwen3-MoE	BitDistiller	53.8	VPTQ	68.7	+14.9

A.4 Cross-paradigm 2-bit comparison

Table 10 compares the trained scalar checkpoints directly with the strongest available vector PTQ result on the same model in AvgQA. On Phi-4-mini the vector advantage is 10.4 points over GSQ and 10.0 points over BitDistiller; on Llama it is 8.7 points over GSQ and 5.4 points over BitDistiller; and on Qwen3-MoE it is 14.9 points over BitDistiller. AQLM, despite using only one fixed calibration recipe, edges BitDistiller by 0.2 points on Llama. The comparison is deliberately not presented as a compute-matched efficiency result: BitDistiller performs full-model QAT, GSQ performs layer-wise discrete optimization, and the vector methods are no-recovery PTQ.

A.5 Training-compute accounting for the recovery comparison

Table 11 reports the training-compute budget used in our recovery-efficiency comparison. We count the dominant algorithmic FLOPs from the completed run configurations, with $N = 3.84 \times 10^9$ parameters. For GSQ, layer inputs are detached, so each optimization token requires a layer forward and weight/logit backward (approximately $4ND$), not a full-model $6ND$ update; we additionally count one activation-propagation pass and GPTQ initialization. BitDistiller costs approximately $8ND$: frozen-teacher forward plus student forward and backward. This gives $16{,}136$ and $15{,}717$ PFLOP, respectively. The difference is only 2.7% and is smaller than the uncertainty of this accounting; accordingly, we treat the methods as compute-comparable rather than claiming a strict ordering.

YAQA includes the Fisher-Hessian pipeline omitted by a calibration-only count. The completed run collects both input and output covariances for 128 linear maps. Besides model forward/backward passes, each covariance contributes an outer product; using the actual Phi-4 layer dimensions gives $2\sum_\ell (d_{\text{in},\ell}^2 + d_{\text{out},\ell}^2) = 2.617 \times 10^{10}$ FLOPs per token. We collected four domain Hessians (code, instruction, reasoning, and web) concurrently on one 8-GPU node and reused to produce eight base/mixture checkpoints. Their observed mean sequence lengths (406, 216, 3455, and 934 tokens) give 426 PFLOP in total, or 53 PFLOP amortized per output. A standalone reason+web checkpoint charged both source Hessians would instead cost approximately 373 PFLOP. The plot uses the amortized value because that is how we produced the reported checkpoint suite. The headline "half the compute of QAT" comparison contrasts the recovery stages directly (PTQ+RFT $8{,}349$ vs BitDistiller $15{,}717$ PFLOP) and is therefore unaffected by this base-quantization Hessian-amortization choice. Matrix factorization and trellis search add less than 0.1 PFLOP at these dimensions. For AQLM we count only the end-to-end codebook fine-tuning over its 128×2048-token reason+web calibration set at $6ND$ per token, giving 6.0 PFLOP per epoch and 151 PFLOP at 25 effective epochs (†); this excludes the K-means/beam codebook-fitting search, whose cost is implementation-dependent and better measured from GPU-hours, so the AQLM entry is a fine-tuning lower bound.

The same accounting extends to the other model families. Table 12 lists the audited per-method training compute for Phi-4-mini, Llama-3.1-8B and Qwen3-30B-A3B; the ordering is stable across models; rotation and forward-only vector PTQ cost tens to hundreds of PFLOP, whereas the learning-based scalar baselines cost over $10{,}000$ PFLOP. For the MoE BitDistiller entry we count activated-parameter passes ($N = 3.34 \times 10^9$): the reported Qwen3-30B-A3B step-800 checkpoint processes $800 \times 64 = 51{,}200$ sequences of $1{,}024$ tokens, so $8ND \approx 1{,}401$ PFLOP.

Table 11: **Compute and weight-memory accounting for the recovery comparison** (Phi-4-mini). FLOPs include the dominant operations described above. Weight GB is the deployed quantized-weight footprint implied by effective bits/weight; it excludes shared non-quantized tensors and runtime buffers. AvgQA is the 7-task average.

Method	Repr.	FLOPs (PFLOP)	Weight GB	AvgQA
SpinQuant 4-bit	scalar	2	1.920	69.2
SpinQuant 2-bit	scalar	2	1.020	29.3
YAQA 2-bit	vector	53	1.085	61.8
AQLM 2-bit	vector	151†	0.960	57.3
PTQ+RFT 2-bit	vector	8,349	1.085	**63.8**
BitDistiller 2-bit	scalar	15,717	1.020	53.0
GSQ 2-bit	scalar	16,136	1.020	52.6

Table 12: **Audited training compute per method (PFLOP).** One-time algorithmic cost from the completed run configurations (§A.5), counting the dominant model passes. Scalar rotations (SpinQuant) and forward-only vector PTQ (VPTQ, YAQA) are orders of magnitude cheaper than the learning-based baselines (GSQ, BitDistiller). *Per token*, BitDistiller's full-model forward+backward ($8N$) costs twice GSQ's block-wise, detached-input update ($4N$); the two nonetheless reach comparable dense-model totals because GSQ optimizes a 0.1B-token calibration set for 10 epochs whereas BitDistiller trains a single epoch over the recovery mixture ($\sim$500k sequences for the dense models). The Qwen3-MoE BitDistiller entry is likewise one epoch, over its smaller 47,616-sequence mixture (the step-800 checkpoint that produced its accuracy row); its budget is correspondingly smaller. PTQ+RFT adds a codebook-only recovery pass (frozen backbone, $6N$/token) to the YAQA base and remains cheaper than BitDistiller on every model ($1.9\times$ on Phi-4-mini, $3.7\times$ on Llama, $54\times$ on Qwen3-MoE). MoE rows use activated-parameter passes. Dashes: configuration not run for that model.

Method	Phi-4-mini	Llama-3.1-8B	Qwen3-MoE
SpinQuant (rotation)	2	4	2
VPTQ (vector PTQ)	36	69	36
YAQA (vector PTQ)	53	728	15
PTQ+RFT (codebook recovery)	8,349	8,938	24
GSQ (scalar QAT)	16,136	33,743	14,035
BitDistiller (QAT)	15,717	32,866	1,401

A.6 Measured deployment and training-cost profile

Table 13 contrasts the methods along the axes that the representation actually changes: storage (on-disk and packed footprints), the peak GPU memory the quantization or recovery procedure needs, and the one-time training compute. Three trends stand out. First, the training-free methods (scalar SpinQuant and forward vector PTQ, VPTQ/YAQA) only need the BF16 model resident, so their peak memory equals a forward pass (11–63 GB across our models). Second, the learning-based baselines diverge sharply: GSQ refines one transformer block at a time and is the *lowest*-memory method (2–17 GB), whereas BitDistiller runs full QAT at $\sim 14P$ and is the highest (54–427 GB, exceeding one 80 GB GPU on the MoE). Third, our PTQ+RFT recovery adds a frozen-backbone codebook pass whose peak memory stays within 1.05–$1.14\times$ of the YAQA base (12–72 GB), while remaining far cheaper in compute than QAT (Table 12). Runtime throughput and bandwidth are architecture-bound (the harness dequantizes every checkpoint to BF16), so we report the deployable low-bit footprint via the Packed column rather than a dense-runtime measurement.

A.7 Complete cross-model calibration sweep

Table 14 retains recipes evaluated on all three main model families. We exclude model-specific mixtures from the matrix to avoid interpreting missing runs as poor results. Over task-complete rows, the QA/Gen ranges are 7.5/26.7 for Phi-4-mini, 4.9/17.0 for Llama, and 3.9/13.5 for Qwen3-MoE.

Table 13: **Per-method cost profile for all 2- and 4-bit baselines.** Effective bits/weight (*Eff.*), on-disk checkpoint size, theoretical *Packed* low-bit footprint (BF16 embeddings kept), *peak GPU memory during the quantization/recovery procedure*, and one-time *Train PFLOP* (from Table 12). Peak memory is the dominant resident state: forward PTQ (SpinQuant, VPTQ, YAQA) holds the BF16 model, so its peak equals the measured forward peak; GSQ refines one transformer block at a time; BitDistiller runs full QAT ($\sim 14P$); PTQ+RFT adds a memory-cheap frozen-backbone recovery pass on top of YAQA. GSQ/QAT/RFT values are the analytic peak-state accounting of §3.3 (full-QAT on Qwen3-MoE exceeds one 80 GB GPU and is sharded in production). On-disk[‡] is a dense-BF16 serialization estimate (SpinQuant, BitDistiller, and AQLM materialize dense BF16; GSQ saves compressed). [§]AQLM peak memory is the materialize-and-fine-tune resident estimate (batch-1 codebook fine-tuning). AQLM's Train PFLOP is left blank: its beam-search codebook optimization does not fit the model-pass accounting used for the other rows and was not separately timed.

		Eff. b/w	On-disk (GB)	Packed (GB)	Peak mem (GB)	Train PFLOP
Phi-4-mini (3.84B)	SpinQuant 4-bit	4.25	7.67[‡]	2.94	11.0	2
	SpinQuant 2-bit	2.12	7.67[‡]	2.08	11.0	2
	GSQ 2-bit	2.12	4.20	2.25	2.2	16,136
	BitDistiller 2-bit	2.12	7.67	2.08	53.8	15,717
	YAQA 2-bit	2.26	9.82	2.31	12.2	53
	PTQ+RFT 2-bit	2.26	9.82	2.31	12.2	8,349
Llama-3.1-8B (8.03B)	SpinQuant 4-bit	4.25	16.06[‡]	5.04	17.7	4
	SpinQuant 2-bit	2.12	16.06[‡]	3.04	17.7	4
	GSQ 2-bit	2.12	1.99[‡]	3.04	5.0	33,743
	BitDistiller 2-bit	2.12	16.06[‡]	3.04	112.4	32,866
	AQLM 2-bit	2.00	16.06[‡]	2.93	17.7[§]	–
	VPTQ 2-bit	2.00	19.79	2.93	17.7	69
	YAQA 2-bit	2.26	18.04	3.17	17.7	728
	PTQ+RFT 2-bit	2.26	18.04	3.17	18.8	8,938
Qwen3-30B-A3B (30.5B, MoE)	SpinQuant 4-bit	4.25	61.06[‡]	16.68	62.8	2
	SpinQuant 2-bit	2.12	61.06[‡]	8.65	62.8	2
	GSQ 2-bit	2.12	8.03[‡]	8.65	17.0	14,035
	BitDistiller 2-bit	2.12	61.06[‡]	8.65	427.0	1,401
	AQLM 2-bit	2.00	61.06[‡]	8.18	62.8[§]	–
	VPTQ 2-bit	2.00	95.55	8.18	62.8	36
	YAQA 2-bit	2.26	69.70	9.16	62.8	15
	PTQ+RFT 2-bit	2.26	69.70	9.16	71.5	24

A.8 CODEBOOK-ONLY RECOVERY ABLATIONS

The codebook-only ablation uses the Phi-4-mini 2-bit YAQA `rw64` checkpoint. The reference configuration is forward KL, $\tau = 1$, full-vocabulary distillation, a BF16 teacher, and web recovery data. Each block varies one factor. Because this ablation fixes a single `rw64` base, its web/forward-KL reference (GSM8K 49.6) differs from the per-recipe web base in Table 5 (GSM8K 40.5); the two should be read within their own base, not compared row-for-row. The report does not contain a top-p run, so we do not include one. The final row is the best validated run, not an untested combination of independently optimal choices.

A.9 ADDITIONAL SCALAR PTQ BASELINES

GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) reach the random-accuracy region at 2 bits. SqueezeLLM (Kim et al., 2024) retains partial capability on the six-task aggregate but nearly zero generation. GSQ (Dadgarnia et al., 2026), despite optimizing its discrete scalar assignments, also remains at the random-accuracy floor in these local Llama runs. These comparisons use the same Llama-3.1-8B evaluation pipeline as the other scalar baselines. Core6 is not directly comparable with AvgQA.

Table 14: **Task-complete 2-bit YAQA calibration sweep.** QA and generation means from rows with complete per-task evaluations. Blank weight entries mean that the source is absent; dashes in result columns mean that only aggregate AvgQA/HEval, not the full task row, was reported. Δ_{complete} excludes dashed result cells.

		Hessian mixture weight				Phi-4-mini		Llama-3.1-8B		Qwen3-30B-A3B	
Group	Recipe	Code	Web	Reason	Instr	QA	Gen	QA	Gen	QA	Gen
Single	Web		1			**63.4**	12.2	65.1	9.6	66.0	54.0
	Instruct				1	56.8	10.7	63.9	11.8	64.9	49.8
	Reasoning			1		58.5	37.4	62.1	**26.6**	64.5	61.8
	Code	1				55.9	18.7	60.8	22.6	62.8	48.9
Standard mix	reason+web		0.5	0.5		62.6	24.7	**65.7**	17.8	**66.7**	56.5
	webreason		0.7	0.3		63.0	27.7	65.5	15.1	65.8	56.2
	web+instruct		0.6		0.4	58.7	14.6	**65.7**	16.5	66.6	51.4
	reason+instruct			0.6	0.4	57.3	18.2	64.6	17.6	64.2	53.8
	code+web	0.6	0.4			58.0	18.2	64.3	21.6	65.2	59.4
	code+reason	0.6		0.4		56.2	19.4	61.6	23.5	63.9	53.7
	code+reason+web	0.4	0.3	0.3		57.8	23.4	64.6	22.0	65.9	61.8
	all-4	0.25	0.25	0.25	0.25	58.8	19.9	65.2	20.1	66.0	59.3
Targeted	rwi442		0.4	0.4	0.2	59.5	17.1	62.7	12.2	66.5	**62.4**
Δ_{complete}						7.5	26.7	4.9	17.0	3.9	13.5

Table 15: **Codebook-only recovery ablations.** Per-task accuracy on Phi-4-mini 2-bit YAQA. AvgQA is the mean of the seven QA tasks; GSM8K and HEval are reported separately.

Axis	Setting	MMLU	WG	ARC-e	ARC-c	HS	LMB	PIQA	GSM8K	HEval	AvgQA
References											
	BF16 ceiling	68.7	71.9	80.3	58.9	72.7	66.0	78.3	81.9	64.6	71.0
	2-bit YAQA, no recovery	51.7	63.6	73.6	48.5	61.7	59.8	73.7	44.9	8.5	61.8
Loss											
	Forward KL	53.7	66.1	75.0	50.4	63.9	60.5	75.1	49.6	19.5	63.5
	Reverse KL	53.6	65.7	74.7	50.4	63.8	60.1	74.9	50.2	20.1	63.3
	Cross-entropy	53.7	65.8	75.0	50.5	63.9	60.5	74.9	49.7	20.1	63.5
Teacher precision											
	BF16	53.7	66.1	75.0	50.4	63.9	60.5	75.1	49.6	19.5	63.5
	4-bit matched	53.7	66.5	74.6	50.4	63.8	61.2	75.1	48.7	18.9	63.6
Temperature											
	$\tau = 0.5$	53.4	66.4	75.0	50.7	63.8	60.3	74.8	48.4	19.5	63.5
	$\tau = 1.0$	53.7	66.1	75.0	50.4	63.9	60.5	75.1	49.6	19.5	63.5
	$\tau = 2.0$	53.7	66.1	74.8	50.3	63.5	60.1	74.9	50.2	19.5	63.3
Logit support											
	$k = 128$	53.6	65.7	75.0	50.4	63.9	60.7	75.1	49.4	20.1	63.5
	$k = 512$	53.6	65.8	75.0	50.5	64.0	60.5	75.0	49.1	20.1	63.5
	$k = 1024$	53.6	65.6	75.1	50.6	63.9	60.6	74.9	49.8	20.1	63.5
	Full vocabulary	53.7	66.1	75.0	50.4	63.9	60.5	75.1	49.6	19.5	63.5
Recovery data											
	Web	53.7	66.1	75.0	50.4	63.9	60.5	75.1	49.6	19.5	63.5
	Instruct	53.2	66.7	75.0	50.2	63.9	59.8	75.1	49.4	14.6	63.4
	Reasoning	53.6	65.3	73.2	48.7	63.3	60.0	74.2	58.4	10.4	62.6
	Code	53.4	66.0	73.0	49.4	63.4	62.0	74.5	53.3	21.3	63.1
	Mixture	53.7	66.1	74.8	50.3	64.0	60.3	74.8	55.0	17.7	63.4
Selected run (best QA/generation balance): mixture data		53.7	66.1	74.8	50.3	64.0	60.3	74.8	55.0	17.7	**63.4**

A.10 BEST CALIBRATION RECIPES

This appendix reports, for every method and model, the *best* 2-bit checkpoint over the calibration recipes we evaluated, selecting QA and generation independently, and records which recipe was selected in each case. We also record the per-metric best recipe for each method.

Selection protocol. For every (model, method) pair we evaluate a set of calibration recipes: single domains (`web`, `instruction`, `reasoning`, `code`, `mixed`) and convex mixtures over them; and, for the trained scalar baselines, a larger `Mixed-500k` corpus. We then pick the recipe with the highest AvgQA for Table 17a and, separately, the highest AvgGen for Table 17b. AQLM is the only exception: it is run once under a fixed $50/50$ reasoning/web recipe, so its QA and generation entries share a single checkpoint.

Table 16: **Named 2-bit scalar PTQ baselines on Llama-3.1-8B.** Core6 averages ARC-Easy, ARC-Challenge, PIQA, HellaSwag, WinoGrande, and LAMBADA; it excludes MMLU and is not the paper's AvgQA metric.

Method	Core6 QA	GSM8K	HEval
GPTQ	30.2	0.0	0.0
AWQ	29.9	0.0	0.0
SqueezeLLM	42.1	1.3	1.2
GSQ (best domain)	30.2	0.0	0.0
Random chance	29.2	0.0	0.0

Table 17: **Cross-paradigm comparison at 2 bits.** This reproduces the main-paper Table 3 with the per-metric best-recipe selection made explicit (Table 18). Best 2-bit AvgQA (a) and AvgGen (b) per model, each selecting the best calibration recipe for its metric independently (Table 18). GSQ is scalar PTQ with layer-wise discrete optimization; BitDistiller is full-model QAT with self-distillation; AQLM, VPTQ, and YAQA are vector PTQ. AQLM uses one fixed $50/50$ reasoning/web recipe. Bold marks the best per model; dashes denote unavailable evaluations.

(a) QA (AvgQA ↑).

		Trained scalar		Vector PTQ		
Model	BF16	GSQ	BitDist.	AQLM	VPTQ	YAQA
Phi-4-mini	71.0	52.6	53.0	57.3	53.9	**63.3**
Llama-3.1-8B	73.1	57.0	60.3	60.5	59.8	**65.7**
Qwen3-MoE	72.6	29.4	55.9	**71.2**	68.7	66.7
OLMoE-1B-7B	68.3	29.6	49.1	**65.5**	65.2	64.0

(b) Generation (AvgGen ↑).

		Trained scalar		Vector PTQ		
Model	BF16	GSQ	BitDist.	AQLM	VPTQ	YAQA
Phi-4-mini	73.2	21.6	18.4	27.0	11.1	**37.4**
Llama-3.1-8B	42.9	20.0	16.2	17.9	16.1	**26.6**
Qwen3-MoE	61.8	0.0	45.7	61.2	58.9	**62.4**
OLMoE-1B-7B	32.4	0.0	0.9	**21.6**	21.4	20.6

QA is recipe-stable; generation is recipe-sensitive. Across YAQA recipes the QA average spans only 3.9–7.5 points while the generation average spans 13.5–25.2 points (Figure 2), so recipe choice moves generation far more than QA. Consequently the QA-optimal recipe is consistently a broad, general corpus, `web` or the $50/50$ reasoning/web mixture on the dense and MoE models, whereas the generation-optimal recipe shifts toward task-aligned, reasoning- and code-heavy data. On both dense models and Qwen3-MoE, VPTQ's generation peaks under `code` calibration, and YAQA's generation peaks under reasoning-heavy mixtures (`w/r .7/.3` on Phi-4-mini, `reasoning` on Llama-3.1-8B, and a reasoning/web/instruction mixture on Qwen3-MoE). The QA winner and generation winner therefore rarely coincide.

A robust default. Because QA is nearly flat in the recipe and generation prefers reasoning-aligned data, the $50/50$ reasoning/web mixture is a strong single-recipe default: it is the QA optimum for YAQA on Llama-3.1-8B and Qwen3-MoE and stays within roughly a point of the best QA elsewhere, while retaining most of the generation benefit of reasoning-heavy calibration. This matches the main-paper recommendation to treat calibration as a generation knob and to blend a small, task-aligned Hessian mixture rather than tuning a single narrow domain. The trained scalar baselines (GSQ, BitDistiller) instead depend on *scale*: their base-domain checkpoints collapse to near-random AvgQA, and only the enlarged `Mixed-500k` / domain-mixture corpora recover usable accuracy, which is the recipe we report for them.

A.11 Trained scalar baseline details

We additionally evaluate two methods that optimize quantized weights: BitDistiller (Du et al., 2024), a group-128 QAT method with self-distillation, and GSQ, a group-128 scalar method that learns discrete assignments after GPTQ initialization. These methods consume training or layer-optimization compute and are therefore not direct no-recovery PTQ baselines. We report only task-complete checkpoints in Table 19. Phi-4-mini BitDistiller and GSQ are the mixed-500k checkpoints, consistent with the headline Table 3. The Qwen3-30B-A3B GSQ checkpoints instead collapse to the random-accuracy floor (29.4 AvgQA, near-zero generation), matching the OLMoE GSQ collapse, so we do not report a recovered GSQ row for that model.

BitDistiller QAT configuration. Our BitDistiller runs apply asymmetric autoclip, convert every quantized projection to a 2-bit group-128 QuantLinear with a Hadamard input rotation, and fine-

Table 18: **Selected calibration recipe per method.** The recipe that maximizes AvgQA and, separately, AvgGen in Tables 17a–17b. Single words denote base domains; fractions denote convex mixtures (e.g. `r/w .5/.5` is 0.5 reasoning + 0.5 web). `Mixed-500k` is the enlarged trained-baseline corpus. Dashes denote methods not run on a model.

Model	Method	Best-QA recipe	Best-Gen recipe
Phi-4-mini	GSQ	Mixed-500k	Mixed-500k
	BitDistiller	Mixed-500k	Mixed-500k
	AQLM	r/w .5/.5	r/w .5/.5
	VPTQ	instruction	code
	YAQA	web	reasoning
Llama-3.1-8B	GSQ	Mixed-500k	Mixed-500k
	BitDistiller	mixture	mixture
	AQLM	r/w .5/.5	r/w .5/.5
	VPTQ	instruction	code
	YAQA	r/w .5/.5	reasoning
Qwen3-MoE	BitDistiller	mixture	mixture
	VPTQ	instruction	code
	YAQA	r/w .5/.5	r/w/i .4/.4/.2
OLMoE-1B-7B	AQLM	r/w .5/.5	r/w .5/.5
	VPTQ	web	w+r
	YAQA	w+r	w+r

Table 19: **Task-complete trained 2-bit baselines.** BitDistiller and GSQ use 2-bit, group-128 scalar weights. Results use the same task metrics as the main evaluation. "Best complete" selects over complete calibration-domain checkpoints only.

Model	Method	Domain	MMLU	WG	ARC-e	ARC-c	HS	LMB	PIQA	GSM8K	HEval	AvgQA	AvgGen
Phi-4-mini	BitDistiller	Mixed-500k	39.2	59.8	66.3	39.7	51.9	45.0	69.3	20.3	16.5	53.0	18.4
Phi-4-mini	GSQ	Mixed-500k	41.0	60.1	63.3	40.2	47.2	47.0	69.2	31.5	11.6	52.6	21.6
Llama-3.1-8B	BitDistiller	Mixture	43.9	66.1	70.2	42.2	65.3	59.7	75.0	9.8	22.6	60.3	16.2
Llama-3.1-8B	GSQ	Mixed-500k	41.1	65.4	67.6	38.6	57.3	56.6	72.4	25.3	14.6	57.0	20.0
Qwen3-30B-A3B	BitDistiller	Mixture	48.8	57.6	62.8	42.0	62.8	43.6	73.8	52.3	39.0	55.9	45.7

tune the student with CAKLD self-distillation against the frozen BF16 teacher. Optimization uses AdamW at learning rate 2×10^{-5} with a cosine schedule, 10% warmup, gradient accumulation 8, per-device batch size 1 across an eight-GPU H100 node, a $1{,}024$-token sequence length, gradient clipping at 1.0, and one epoch over the calibration mixture (seed 42). Qwen3-30B-A3B is FSDP-sharded to fit optimizer state on a single node; its reported step-800 checkpoint corresponds to $51{,}200$ training sequences (§A.5).

Calibration remains consequential for trained baselines (Table 20). The strongest complete Llama BitDistiller run is the later mixture checkpoint, which improves AvgQA by 4.2 points and HumanEval by 20.8 points over its web checkpoint. The five base GSQ domains collapse to roughly 30–32 AvgQA, but a larger mixed-500k GSQ checkpoint recovers to 57.0 AvgQA and 14.6 HumanEval, which is the row we report in Table 19. We do not merge newer MMLU-only retries with older task-complete checkpoints because doing so would create synthetic rows from different model artifacts.

A.12 AQLM BASELINE DETAILS

AQLM (Egiazarian et al., 2024) provides a named modern vector-quantization baseline. It uses one fixed 50/50 reasoning/web calibration rather than a recipe sweep. On the two completed dense models, the matched YAQA results (62.6 on Phi-4-mini and 65.7 on Llama), as well as the best YAQA AvgQA results (63.0 and 65.7), exceed AQLM, while AQLM

Table 21: **AQLM 2-bit results.** A dash denotes an incomplete evaluation.

Model	AvgQA	HEval	IFEval
Phi-4-mini	57.3	9.8	24.6
Llama-3.1-8B	60.5	12.2	10.9
OLMoE-1B-7B	65.5	19.5	25.9
Qwen3-MoE	71.2	34.8	21.6

Table 20: **Calibration-domain sensitivity of trained 2-bit baselines on Llama-3.1-8B.** AvgQA and HumanEval. BitDistiller and GSQ report their five base-domain checkpoints plus the larger mixed-500k recipe.

Method	Domain	AvgQA	HEval
BitDistiller	Web	56.1	1.8
	Instruct	35.9	0.0
	Reasoning	38.0	0.0
	Code	32.4	0.6
	Mixture	**60.3**	**22.6**
	Mixed-500k	54.9	15.9
GSQ	Web	31.3	0.0
	Instruct	32.4	14.6
	Reasoning	29.9	11.6
	Code	29.7	14.0
	Mixture	31.7	14.6
	Mixed-500k	**57.0**	**14.6**

Table 22: **QTIP (single-Hessian) versus YAQA (two-sided Hessian).** Per-domain 2-bit AvgQA (seven-task mean); same codebook and rotation, only the rounding Hessian differs. Δ is QTIP minus YAQA. The two-sided Hessian helps Phi-4-mini but not Llama.

Model	Calib. domain	QTIP	YAQA	Δ
Phi-4-mini	web	58.3	63.3	-5.0
	instruction	53.7	56.8	-3.1
	reasoning	54.4	58.5	-4.1
	code	51.0	55.9	-4.9
Llama-3.1-8B	web	65.2	65.1	$+0.1$
	instruction	65.3	63.9	$+1.4$
	reasoning	63.1	62.1	$+1.0$
	code	62.4	60.8	$+1.6$

remains much stronger than the scalar methods in Table 16. On Qwen3-30B-A3B, AQLM reaches 71.2 AvgQA, 34.8 HumanEval, and 21.6 prompt-strict IFEval.

A.13 QTIP: THE VALUE OF THE TWO-SIDED HESSIAN

YAQA's trellis codebook is shared with QuIP#/QTIP-style quantizers (Tseng et al., 2024a;b); the difference is the rounding objective. YAQA rounds with a two-sided Fisher Hessian (input and output), whereas QTIP uses a single input-side Hessian. We isolate exactly this change (same trellis codebook, same incoherence rotation, and the same per-domain Hessians) and evaluate a 2-bit QTIP variant. Table 22 shows the two-sided Hessian is what drives YAQA's dense-model gains, and that its benefit is model-dependent: it adds 3–5 AvgQA points per calibration domain on Phi-4-mini but is roughly neutral on Llama-3.1-8B. QTIP also nearly eliminates code generation on Phi-4-mini (HumanEval ≤ 4.9 across domains, versus 18–21 for YAQA), so the single-Hessian variant is not a substitute for YAQA on the harder-to-quantize dense model.

A.14 MOE ROUTER PRECISION

Quantizing the Qwen3-MoE router does not explain the model's strong 2-bit retention. Across five checkpoints, 2-bit routers stay within 0.2 points of fp16 routers.

A.15 LORA-AUGMENTED RECOVERY

Adding LoRA (Hu et al., 2022) adapters on top of codebook recovery increases attainable generation accuracy, but also changes the trainable and stored parameterization. We therefore report each checkpoint separately rather than combining per-metric maxima. All HumanEval scores use adapters folded into the materialized weights. On Phi-4-mini, AvgQA is stable across recovery data (63.6–64.4); the specialization is in generation, where reasoning gives the strongest GSM8K (63.3)

Table 23: **Qwen3-MoE router-precision ablation.** Expert weights remain at their vector-quantized 2-bit setting; only router precision changes. The checkpoint labels are those in the router-ablation report and their absolute scores differ from the headline domain sweep; only within-row precision deltas should be compared.

Checkpoint	fp16	8-bit	4-bit	2-bit
VPTQ instruction	65.5	65.5	65.6	65.4
VPTQ web	67.0	66.9	67.1	67.1
VPTQ code	64.0	64.0	64.1	64.0
YAQA reason+web	68.5	68.5	68.5	68.6
YAQA RWI442	68.2	68.1	68.0	68.0

Table 24: **LoRA + codebook recovery by recovery dataset.** Per-task accuracy after folding the adapters into each materialized checkpoint. AvgQA averages the seven QA tasks. The Llama instruction row is omitted because only HumanEval (20.1) is available. Phi-4-mini aggregates cover all reported configurations.

Model	Recovery data	MMLU	WG	ARC-e	ARC-c	HS	LMB	PIQA	GSM8K	AvgQA	HEval
Phi-4-mini	Web	55.8	65.7	76.3	51.0	65.5	61.5	75.3	45.3	**64.4**	17.7
	Instruct	55.6	65.3	75.5	51.0	65.7	60.7	75.5	50.7	64.2	30.5
	Reasoning	54.8	64.3	76.3	50.0	65.1	60.5	74.4	**63.3**	63.6	19.5
	Code	55.1	64.6	73.7	50.4	65.4	61.8	75.0	52.5	63.7	**32.9**
	Mixture	56.0	64.6	75.0	50.9	65.0	60.5	75.2	56.9	63.9	29.3
Llama-3.1-8B	Web	49.7	69.9	76.6	48.8	73.4	69.0	78.7	23.3	66.6	13.4
	Reasoning	49.7	69.2	77.4	48.3	73.3	69.2	78.2	**33.7**	66.5	14.6
	Code	48.7	69.1	76.8	48.6	73.7	68.4	78.2	26.1	66.2	**26.2**
	Mixture	50.2	70.3	78.3	48.7	74.5	67.5	78.2	30.9	**66.8**	19.5

and code gives the strongest HumanEval (32.9). The same pattern appears on Llama: AvgQA stays near 66–67 while reasoning maximizes GSM8K (33.7) and code maximizes HumanEval (26.2). The best folded Phi checkpoint reaches 34.8 IFEval prompt-strict accuracy; the evaluation record does not identify that checkpoint's recovery-data row, so we report it only as a best validated outcome rather than attaching it to Table 24.

Full-weight fine-tuning does not help. At the opposite extreme from codebook-only recovery, we also fine-tune *every* 2-bit weight (a full-weight control with WebReason-500k recovery data) on Phi-4-mini. This most-permissive parameterization is markedly *worse*: it recovers only 6.7 HumanEval and 12.8 IFEval, versus 36.6 and 39.2 for codebook-only recovery, with no QA improvement. At 2 bits the limited recovery budget is best spent on the separate codebook rather than on the full weight matrix, reinforcing the codebook-only default.

Table 25 reports the remaining one-factor-at-a-time ablation axes. On Phi-4-mini, changing the loss, temperature, logit support, or teacher precision moves AvgQA by at most 0.4 points, while HumanEval ranges from 15.9 to 21.3. This mirrors codebook-only recovery: optimization details are comparatively stable, whereas recovery-data composition drives the largest task-specific changes. Llama shows the same stability, with AvgQA within 0.3 points across every axis; only the $\tau = 2$ setting was not trained for Llama and is left blank.

A.16 QWEN3-MOE RECOVERY-DATA SWEEP

We additionally run a separate 2-bit Qwen recovery fine-tuning pipeline. It is not the headline VPTQ/YAQA model and does not include MMLU, GSM8K, or HumanEval, so its five-task average, AvgQA5, is not directly comparable to the seven-task AvgQA used elsewhere. Within that controlled sweep, however, broad web and mixture data outperform narrow instruction, reasoning, and code recovery by 16–20 points.

A.17 QWEN3-8B: A THIRD DENSE MODEL

We additionally evaluate Qwen3-8B (Yang et al., 2025), a third dense model, at 2 bits under the reason+web calibration mixture. Consistent with the dense-model ordering in the main paper, YAQA outperforms VPTQ on AvgQA by 2.9 points (63.7 versus 60.8), on generation by 5.5 HumanEval

Table 25: **LoRA + codebook one-factor-at-a-time ablations.** Aggregate results for the non-data axes. The baseline uses forward KL, $\tau = 1$, full-vocabulary logits, a BF16 teacher, and web recovery data. Dashes denote unavailable complete task aggregates.

		Phi-4-mini		Llama-3.1-8B	
Axis	Setting	AvgQA	HEval	AvgQA	HEval
Baseline	Forward KL, $\tau = 1$, full vocab	64.4	17.7	66.6	13.4
Loss	Reverse KL	64.1	**21.3**	66.5	15.2
	Cross-entropy	64.4	19.5	66.6	14.0
Temperature	$\tau = 0.5$	64.1	16.5	66.3	14.0
	$\tau = 2$	64.0	18.9	–	–
Logit support	$k = 128$	64.4	17.7	66.3	14.0
	$k = 512$	64.5	17.7	66.5	13.4
	$k = 1024$	64.5	16.5	66.5	13.4
Teacher	4-bit matched	64.3	15.9	66.3	13.4

Table 26: **Qwen3-MoE recovery fine-tuning data sweep.** AvgQA5 averages ARC-Easy, ARC-Challenge, PIQA, HellaSwag, and WinoGrande; LAMBADA is shown but excluded. MMLU, GSM8K, and HumanEval are not evaluated in this controlled sweep.

Data	ARC-e	ARC-c	PIQA	HS	WG	LMB	AvgQA5
Web	69.1	42.0	70.7	58.4	56.3	37.7	**59.3**
Mixture	67.4	43.4	70.4	56.6	57.6	37.5	59.1
Instruct	34.0	25.6	59.4	44.3	52.6	23.3	43.2
Reasoning	29.4	25.3	55.3	45.1	54.9	17.9	42.0
Code	31.1	23.9	54.8	37.5	51.3	12.9	39.7

points (7.9 versus 2.4), and on nine of the ten evaluated tasks, with ARC-Easy the only exception. Only the reason+web calibration mixture is currently available for this model.

A.18 A SECOND SPARSE MOE

An incomplete norm fusion initially made OLMoE (Muennighoff et al., 2025) appear to collapse. After folding the pre-MoE RMSNorm into all 64 experts, the fp32 invariance check passes and 2-bit VPTQ recovers approximately 95% of the BF16 AvgQA score. We include this result because it guards against attributing a graph transformation bug to quantization.

A.19 INSTRUCTION FOLLOWING

IFEval is sensitive to whether the original checkpoint is instruction-tuned. The Phi-4-mini instruct model has a meaningful ceiling and loses instruction following at 2 bits; codebook-only recovery restores only part of that loss. The base Llama checkpoint has a low IFEval ceiling, so its absolute values should not be compared directly with Phi-4-mini. Across two independent MoE families the 2-bit models remain usable instruction-followers: Qwen3-MoE holds 24.4 (from a BF16 ceiling of 28.3) and OLMoE-1B-7B-instruct reaches 25.9 at 2 bits (AQLM), so instruction following is not a casualty of the 2-bit transition.

A.20 ADDITIONAL ANALYSES

Figures 2–4 distill three cross-cutting trends from our 2-bit sweeps, and Figures 5–7 together with Table 30 add four analyses that reinforce the central claims: multiple-choice QA survives 2-bit vector PTQ while generation is the fragile axis (Figure 5), the two axes separate dense from MoE models (Figure 6), vector PTQ lies on the bits-per-weight Pareto frontier (Figure 7), and the scalar 2-bit cliff is universal across rotation methods (Table 30).

Table 27: **Qwen3-8B (dense) at 2 bits, reason+web calibration.** Per-task accuracy; AvgQA averages the seven QA tasks. HumanEval reports pass@1. Bold marks the better of VPTQ and YAQA per column.

Method	MMLU	WG	ARC-e	ARC-c	HS	LMB	PIQA	GSM8K	AvgQA	HEval
BF16	75.0	67.8	80.9	56.4	74.9	64.6	77.7	87.3	71.0	64.0
VPTQ	56.7	62.4	**75.7**	46.2	59.6	52.0	73.0	64.4	60.8	2.4
YAQA	**63.7**	**65.7**	74.1	**48.3**	**65.4**	**55.6**	**73.2**	**68.6**	**63.7**	**7.9**

Table 28: **OLMoE-1B-7B-Instruct at 2 bits.** The model has 64 non-fused experts.

Configuration	AvgQA	GSM8K	HEval	PTB ppl
BF16 ceiling	68.3	33.7	31.1	14.7
VPTQ web	65.2	18.2	15.9	18.6
VPTQ web+reason	64.7	22.7	20.1	18.4
YAQA web+reason	64.0	26.0	15.2	19.3

A.21 Generation under the 2-bit cliff

Figure 9 reproduces the main-paper teaser for generation (AvgGen, the mean of GSM8K and HumanEval) on Phi-4-mini. The scalar cliff is even sharper than for QA: SpinQuant generation falls from 65.9 at 4-bit to 3.5 at 3-bit and 0.0 at 2-bit, and a vector codebook (YAQA, rw64 base) restores 26.7 at 2-bit (left). Under a fixed 2-bit budget, data-centric recovery (PTQ+RFT on the 2-bit vector model, rw64 recommended checkpoint) reaches 43.2 AvgGen at a fraction of the estimated compute of trained baselines such as GSQ (21.6) and BitDistiller (18.4) (right). These panels use the same checkpoints and metric conventions as the QA teaser.

A.22 Accuracy and generation vs. training compute (other models)

Figures 10 and 11 extend the Figure 1(b) view to Llama-3.1-8B and the Qwen3-30B-A3B MoE: each 2-bit method is plotted at its Table 3 score against its audited training compute (Table 12), with the 4-bit and 2-bit scalar ladder for reference. The QA panels reproduce the main ordering, namely vector PTQ (YAQA/VPTQ) matches or beats the learning-based scalar baselines (GSQ, BitDistiller) at 10^2–$10^3\times$ less compute. The generation panels show the architecture split: the dense Llama loses most of its BF16 generation at 2 bits (best vector AvgGen 26.6 versus a BF16 ceiling of 42.9), whereas the Qwen3-MoE retains it (vector AvgGen up to 62.4, essentially lossless).

A.23 Overthinking under quantization

We reproduce the overthinking failure mode of Lotfi et al. (2026) on our own quantized checkpoints. On MATH-500 (Hendrycks et al., 2021b) (levels 4–5, greedy decoding) we compare the BF16 ceiling against the best 4-bit and best 2-bit checkpoint of a reasoning model with an explicit thinking mode (Qwen3-30B-A3B; the best 2-bit is the validated VPTQ `web_mixed_73` checkpoint). Accuracy is exact match on the final `\boxed{}` answer; "markers/1k" counts hesitation and branch-forking tokens ("Wait", "But", "Alternatively", ...) per $1{,}000$ generated tokens; the loop score is the fraction of repeated 4-grams.

Table 32 shows that 2-bit quantization *amplifies* overthinking but does not collapse the model: accuracy falls moderately ($0.675 \rightarrow 0.50$), while the hesitation-marker rate roughly doubles ($15.5 \rightarrow 32.2$ per 1k), the loop score rises ($0.39 \rightarrow 0.58$), and the length-cap hit rate grows ($25 \rightarrow 40\%$). A decoding sanity check confirms this is not a greedy artifact (sampling at $T{=}0.6$ gives 0.60 versus greedy 0.575). The penalty sweep (Table 33) is a *separate* decoding run whose unpenalized greedy baseline (0.575) differs from the headline Table 32 greedy value (0.500) by run-to-run variance on the MATH-500 subset; penalty effects should be read within each run, not across the two tables.

Crucially, the degradation is a decoding pathology rather than a lost capability: a small repetition penalty recovers 2-bit accuracy to the BF16 level (Table 33). A penalty of 1.3 raises 2-bit accuracy from 0.575 to 0.70 while cutting the loop score from 0.54 to 0.26 and shortening the chain-of-

Table 29: **IFEval prompt-strict accuracy.**

Model	Configuration	IFEval
Phi-4-mini-instruct	BF16	56.6
	2-bit YAQA	37.2
	+ codebook recovery	39.2
	2-bit AQLM	24.6
Llama-3.1-8B	BF16	8.9
	2-bit YAQA	7.4
	2-bit AQLM	10.9
Qwen3-30B-A3B	BF16	28.3
	2-bit YAQA	24.4
	2-bit AQLM	21.6
	2-bit VPTQ	15.3
OLMoE-1B-7B-instruct	2-bit YAQA	25.5
	2-bit VPTQ	26.4
	2-bit AQLM	25.9

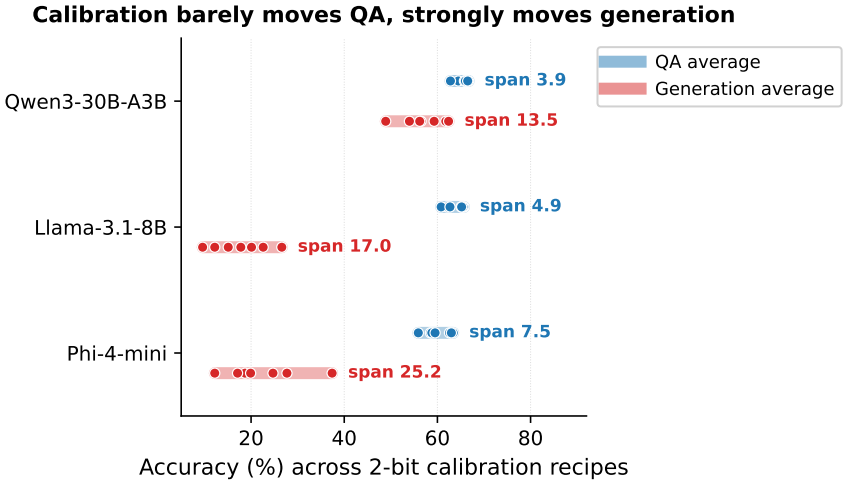


Figure 2: **Calibration composition primarily moves generation, not QA.** Across 2-bit YAQA calibration recipes, the QA average spans only 3.9–7.5 points, whereas the generation average spans 13.5–25.2 points.

thought. The single-domain and mixture checkpoints elsewhere in this paper are unaffected: we always evaluated them on correctly rotated exports.

Qualitative examples. Two representative MATH-500 generations from the best 2-bit Qwen3-MoE checkpoint illustrate the two regimes; Figure 12 shows that a repetition penalty moves the model from the second regime back to the first.

Corrected 2-bit: bounded and correct (Level-4, gold $= \sqrt{5}$)

"I need the radius of a right cylindrical tank filled at 20π *m*^{3}*/hr while the level rises* 4 *m/hr. Since* $V = \pi r^2 h$, $\frac{dV}{dt} = \pi r^2 \frac{dh}{dt}$, *so* $20\pi = \pi r^2 \cdot 4 \Rightarrow r^2 = 5 \Rightarrow r = \sqrt{5}$." The model emits a bounded 1,282-token chain (loop score 0.09) and terminates with the correct $\texttt{\textbackslash boxed}\{\sqrt{5}\}$.

2-bit: looping failure (Level-5, gold $= 6 - 5i$)

"I need to rotate z *around* c *by* $\pi/4$*:* $w = (z - c)e^{i\pi/4} + c$. *Let me compute* $z - c \ldots$" The chain reasons coherently at first but then collapses into a degenerate repetition (loop score 0.89, 1,676 hesitation markers), filling the entire 8,192-token budget *without ever emitting a boxed answer*, so it is scored wrong.

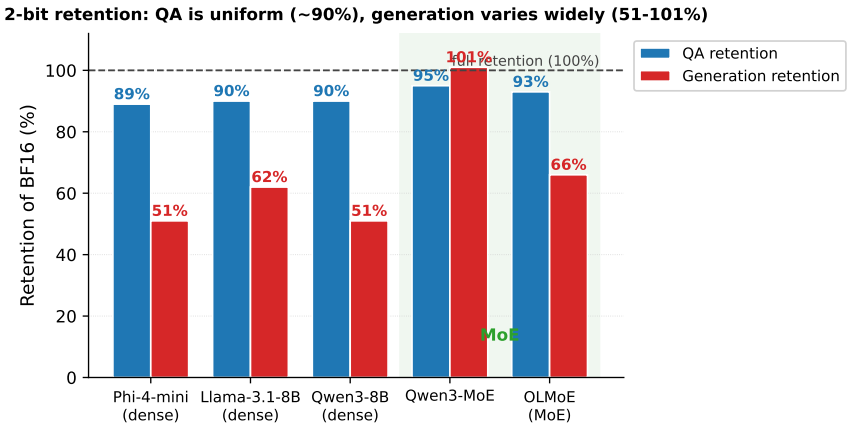


Figure 3: **2-bit retention across models.** QA retention is $\sim 90\%$ for every family, whereas generation retention varies widely (51–101%): only Qwen3-MoE retains generation near-losslessly, while OLMoE (66%) is closer to the dense models.

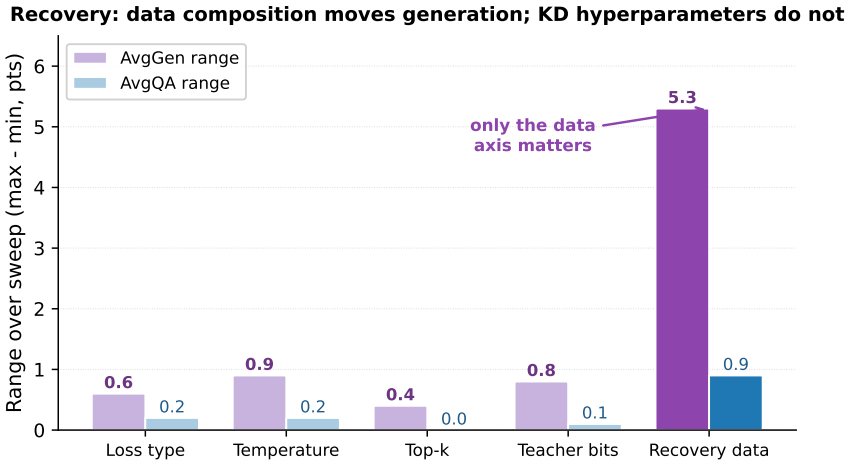


Figure 4: **In recovery, data composition dominates; KD hyperparameters move it little across the tested grid.** Loss type, temperature, top-k, and teacher precision each move AvgGen by ≤ 0.9 and AvgQA by ≤ 0.2; only the recovery-data axis moves generation (AvgGen range 5.3).

A.24 Limitations and unresolved questions

Three limitations bound our conclusions. First, VPTQ and YAQA use different index rates (2.00 versus approximately 2.26 bits/weight), and neither headline rate includes retained fp16 tensors or amortized codebook storage. Second, calibration results select over multiple recipes, so comparisons with single-recipe baselines such as AQLM must remain protocol-matched; the GSQ and BitDistiller comparisons are also not compute-matched. Third, the evaluation does not yet include multi-seed uncertainty, end-to-end memory and latency, or complete recovery coverage on every architecture. Accordingly, we view codebook-only recovery as a useful restricted refinement, not as a substitute for full-precision training or evidence of QAT parity. Its benefit is concentrated in generation: on the dense models it lifts HumanEval substantially while changing AvgQA by only about two points over the no-recovery base, so a compute-matched "more calibration data, no recovery" control is the right way to isolate its QA contribution and is left to future work. Finally, our task-aligned calibration and recovery mixtures draw on the same broad domains as the evaluation task families (reasoning, code), so part of the observed gain may reflect distributional alignment with the benchmarks rather than capability recovery alone; disjoint-corpus controls would separate these effects.

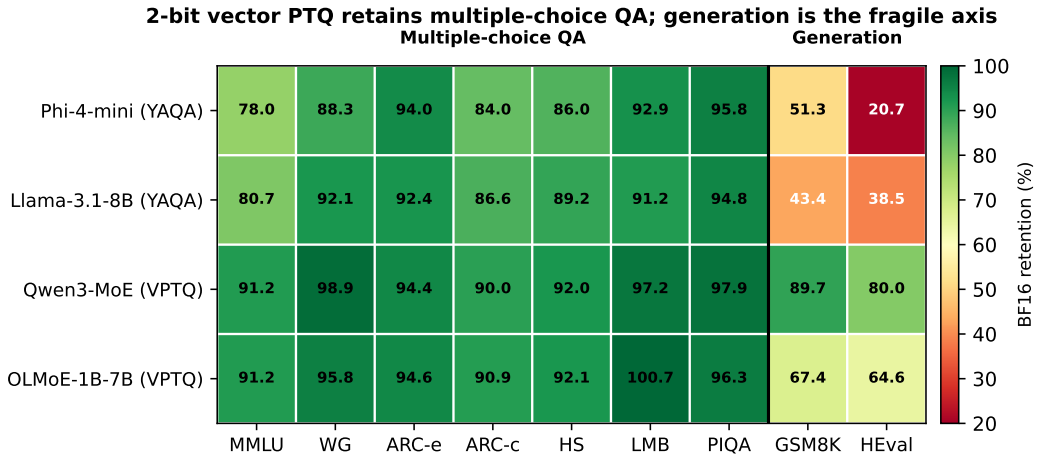


Figure 5: **Which capabilities survive 2-bit vector PTQ.** Per-task retention of the best 2-bit vector checkpoint relative to BF16. Multiple-choice QA is largely retained (84–101% across all four models), whereas generation (GSM8K, HumanEval) is the fragile axis: it falls to 21–51% on the dense models but stays high on the MoE models (65–90%).

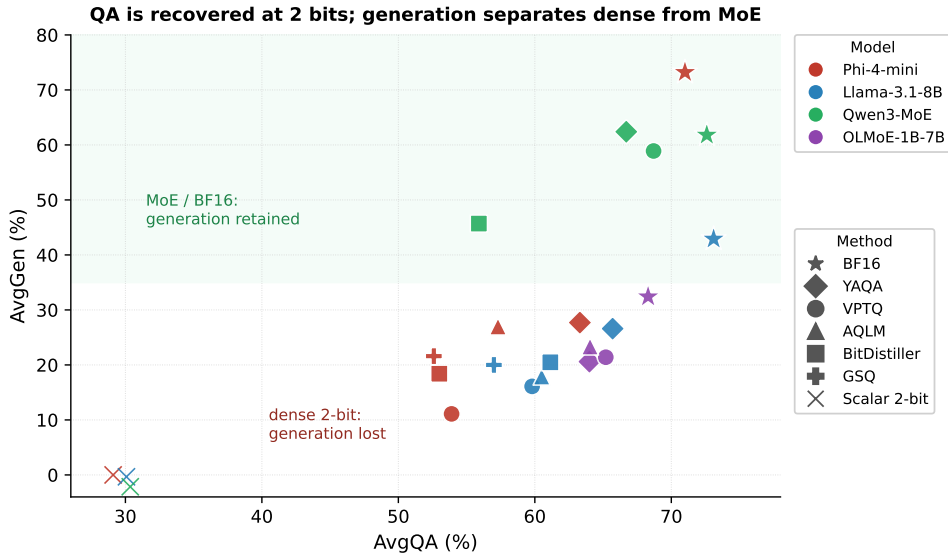


Figure 6: **QA and generation are distinct axes.** AvgQA versus AvgGen for every method and model at 2 bits. Every non-scalar method recovers QA, but generation cleanly separates the dense models (low) from the MoE models and BF16 (high); scalar 2-bit collapses on both axes.

Reporting scope and conventions. Our study covers five checkpoints: three primary models (Phi-4-mini, Llama-3.1-8B, Qwen3-30B-A3B) evaluated under the full protocol, and two auxiliary models (OLMoE, Qwen3-8B) with narrower coverage. We report nominal index rates (2.00 bits/weight for VPTQ and AQLM, ∼2.26 for YAQA); the effective scalar 2-bit rate is $2 + 16/128 = 2.125$ bits/weight with a single FP16 scale per group of 128 (rounded to 2.12 in the tables), rising to 2.25 if an FP16 zero-point is also stored, and total packed size additionally includes codebooks, scales, and retained high-precision tensors. Memory figures should be read by category: deployed packed weights, persistent training state, and measured per-device peak are distinct quantities and are tabulated separately in Table 13. Finally, the decision guide (Table 6) is conditional: the preferred quantizer and recipe depend on the deployment objective (QA, generation, or instruction following), the storage and compute budget, and low-bit kernel availability; we recommend VPTQ for MoEs on storage and latency grounds even though fixed-recipe AQLM attains the strongest MoE QA (Table 3).

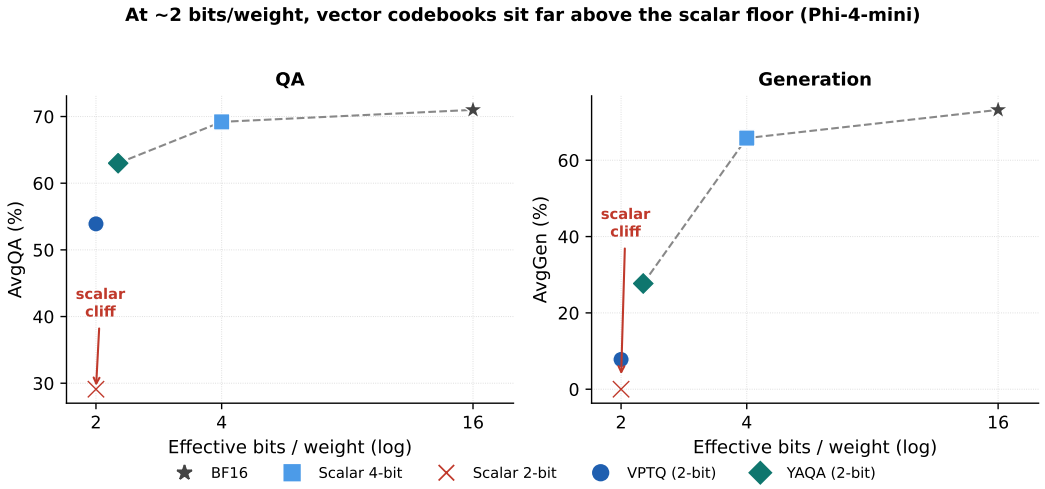
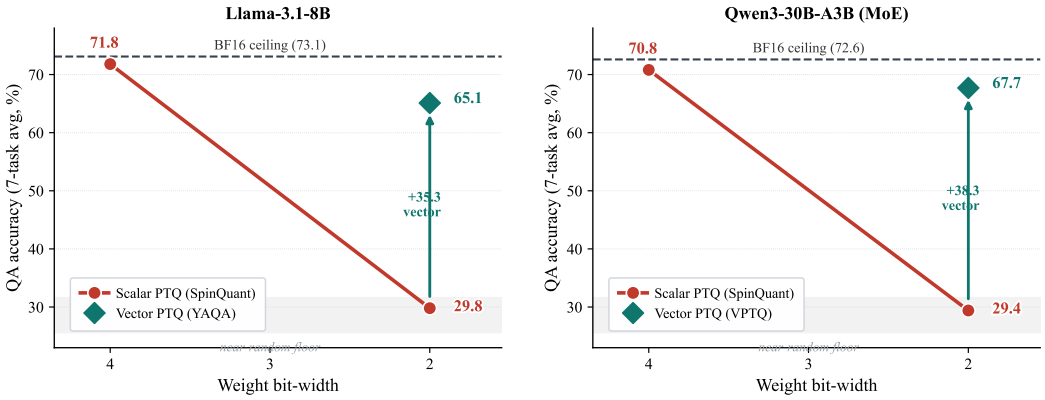


Figure 7: **Representation, not bitwidth, sets the 2-bit frontier.** Effective bits-per-weight versus AvgQA and AvgGen on Phi-4-mini. At $\sim$ 2 bits/weight, a vector codebook (YAQA, 2.26 b/w; VPTQ, 2.00 b/w) sits far above the scalar 2-bit floor and near scalar 4-bit, placing vector PTQ on the accuracy–compression Pareto frontier.

Table 30: **The 2-bit scalar cliff is universal across rotation methods.** Best-calibration-domain AvgQA at 16, 4, and 2 bits for two scalar PTQ methods. Every method stays within 1–2 points of BF16 at 4 bits and collapses to the $\sim$29% random-accuracy floor at 2 bits. QuaRot is applied through expert-wise rotation on Qwen3-MoE.

Model	Method	BF16	4-bit	2-bit
Phi-4-mini	SpinQuant	71.0	69.2	29.3
	QuaRot	71.0	69.3	29.8
Llama-3.1-8B	SpinQuant	73.1	71.8	29.8
	QuaRot	73.1	71.7	29.3
Qwen3-MoE	SpinQuant	72.6	70.8	29.4
	QuaRot	72.6	70.3	29.3

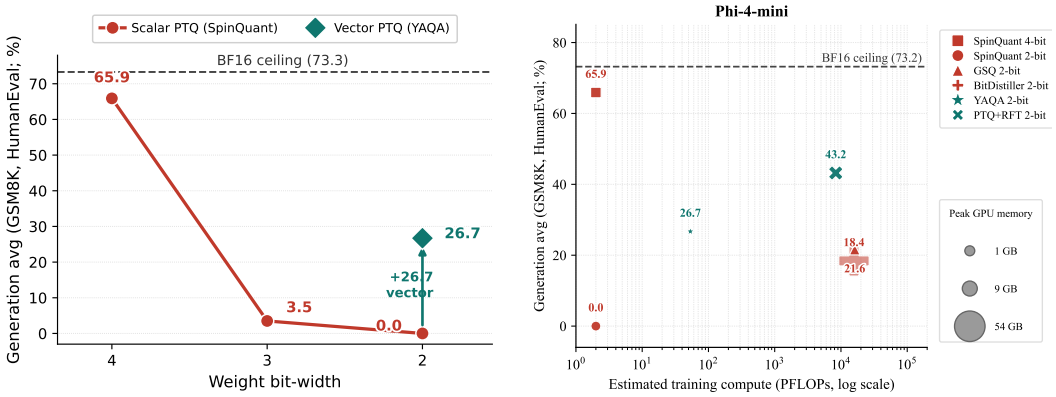


(a) Llama-3.1-8B. (b) Qwen3-30B-A3B (MoE).

Figure 8: **The 2-bit scalar cliff on the other models** (analogs of Figure 1(a)). Scalar PTQ (SpinQuant) AvgQA collapses from near-lossless at 4 bits to the $\sim$ 29% random-accuracy floor at 2 bits, whereas a 2-bit vector codebook (YAQA on Llama, VPTQ on the MoE) recovers most of the gap. AvgQA is the seven-task mean.

Table 31: **Qwen3-MoE 2-bit VPTQ domain-mixture sweep.** Convex-combination calibration mixtures of reasoning (r), web (w), and mixed (m) domains for a single 2-bit VPTQ Qwen3-30B-A3B checkpoint (one codebook per mixture). AvgQA is the seven-task mean; MMLU and GSM8K are 5-shot, HumanEval is pass@1, and IFEval is prompt-level strict. Calibration composition moves generation far more than QA (QA spans 2.7 points; HumanEval spans 21 points). These finer convex mixtures lie *outside* the common single-domain recipe grid selected in Tables 3 and 17; the strongest entries here—AvgQA 69.3 (web/mixed $0.7/0.3$) and AvgGen $(81.8{+}53.0)/2 = 67.4$ (web/reasoning $0.7/0.3$)—exceed those tables' Qwen VPTQ values and, if pooled, would raise VPTQ's best MoE generation above YAQA. We keep the headline tables restricted to the shared single-domain grid for a like-for-like cross-method comparison and report these finer mixtures here.

Calibration mixture	AvgQA	GSM8K	HumanEval	IFEval
reasoning .5 / web .5	67.0	82.0	31.7	23.8
web .7 / reasoning .3	68.4	81.8	**53.0**	22.6
web .5 / mixed .5	67.6	**82.9**	37.8	23.7
web .7 / mixed .3	**69.3**	80.4	48.8	24.8
reasoning .5 / mixed .5	66.6	82.1	44.5	23.7
r .4 / w .4 / m .2	68.0	81.0	47.6	22.0
r .2 / w .4 / m .4	68.3	80.4	43.3	**27.0**
r .3 / w .4 / m .3	68.3	80.0	37.2	25.0



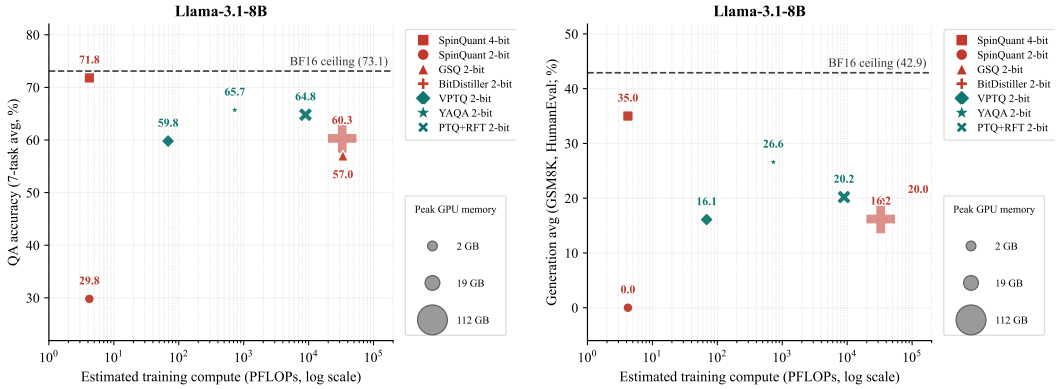
(a) Generation cliff and vector recovery.

(b) Generation vs. estimated training compute at 2 bits.

Figure 9: **Generation mirrors the QA teaser.** (a) Scalar SpinQuant generation collapses by 3-bit and vanishes at 2-bit; a vector codebook restores 26.7 AvgGen. (b) At a fixed 2-bit budget, data-centric recovery attains the highest AvgGen per unit of compute. AvgGen is the mean of GSM8K and HumanEval on Phi-4-mini; marker area is each algorithm's peak GPU memory. Per-method numbers in Table 12.

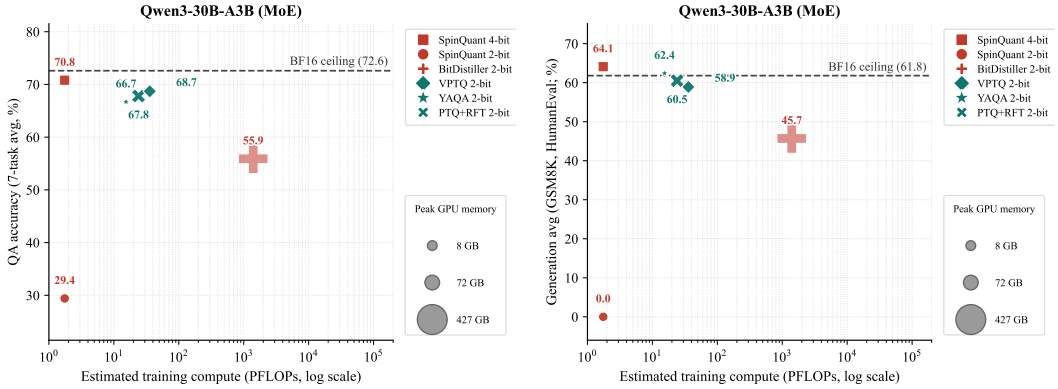
Table 32: **Overthinking under 2-bit quantization** (Qwen3-30B-A3B, MATH-500, greedy). 2-bit amplifies hesitation and looping and moderately lowers accuracy, but does not collapse. "Cap-hit" is the length-cap hit rate; "R-then-A" is the reached-then-abandoned rate among wrong answers.

Precision	Accuracy	Mean CoT	Cap-hit	Markers/1k	Loop (4-gram)	R-then-A
BF16	0.675	4928	0.25	15.5	0.39	0.69
4-bit	0.650	5265	0.25	12.4	0.41	0.50
2-bit	0.500	5396	0.40	32.2	0.58	0.45



(a) AvgQA (7-task). (b) AvgGen (GSM8K, HumanEval).

Figure 10: **Llama-3.1-8B accuracy and generation vs. training compute** (analog of Figure 1(b)). Scalar = crimson, vector = teal; x-axis is audited training compute (log scale); marker area is each algorithm's peak GPU memory. Per-method numbers in Table 12.



(a) AvgQA (7-task). (b) AvgGen (GSM8K, HumanEval).

Figure 11: **Qwen3-30B-A3B (MoE) accuracy and generation vs. training compute** (analog of Figure 1(b)). The MoE retains generation at 2 bits, unlike the dense models. Marker area is each algorithm's peak GPU memory; per-method numbers in Table 12.

Table 33: **A small repetition penalty recovers 2-bit accuracy** (Qwen3-30B-A3B, MATH-500, 2-bit). A repetition penalty of 1.3 restores accuracy to the BF16 level while reducing looping and chain-of-thought length.

Rep. penalty	Marker penalty	Accuracy	Mean CoT	Cap-hit	Markers/1k	Loop (4-gram)
1.0	0	0.575	5078	0.275	19.0	0.54
1.0	2	0.650	4655	0.225	11.1	0.51
1.0	6	0.625	3966	0.200	2.8	0.48
1.2	0	0.675	4398	0.225	15.6	0.33
1.3	0	**0.700**	4256	0.150	12.9	0.26

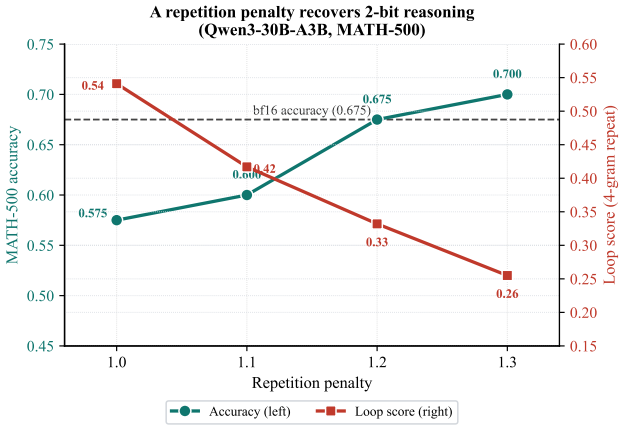


Figure 12: **A repetition penalty recovers 2-bit reasoning** (Qwen3-30B-A3B, MATH-500, 2-bit). Increasing the penalty raises accuracy to the BF16 level ($0.575 \rightarrow 0.70$) while the 4-gram loop score falls ($0.54 \rightarrow 0.26$): the 2-bit degradation is a decoding pathology, not a lost capability.