Statistically-Lossless Quantization of Large Language Models

Michael Helcig
ETH Zürich

Eldar Kurtić
Red Hat AI & ISTA

Dan Alistarh
ISTA

Abstract

Model quantization has become essential for efficient large language model deployment, yet existing approaches present clear trade-offs: methods such as GPTQ and AWQ achieve practical compression but are *lossy*, while *lossless* techniques preserve fidelity but lack inference acceleration. This paper explores the middle ground of *statistically-lossless compression*, examining three complementary aspects of what losslessness means for quantized LLMs. First, *task-lossless* compression preserves zero-shot benchmark accuracy within natural sampling variance and is achievable at aggressive bitwidths. Second, we formalize the stricter notion of *distribution-lossless* compression, requiring the quantized model's next-token distribution to be practically indistinguishable from the original, and propose the *Expected Acceptance Rate (EAR)*, the maximum token-agreement probability under optimal coupling, as a directly interpretable fidelity metric (e.g. EAR ≥ 0.99 means 99% agreement). Third, we prove a *gamma-squared variance law* showing that symmetric quantization inflates noise variance by γ^2 relative to asymmetric, making asymmetric quantization a prerequisite for distribution-lossless fidelity but not for task-level preservation. Through *SLQ*, a layer-wise non-uniform method with asymmetric quantization and wide bitwidth search, we obtain task-lossless compression at well below 4 bits per parameter (as low as 3.3 bits depending on the model), distribution-lossless compression at 5–6 bits per parameter on average, and inference speedups of 1.7–3.7$\times$ compared to FP16 using optimized kernels. Source code is available at `https://github.com/IST-DASLab/SLQ`.

1 Introduction

The large computational and memory costs of large language model (LLM) inference have led *model quantization*, i.e. the reduction in bit-width for weights, activations, or both, to become a standard technique for efficient deployment. As such, over the past few years, hundreds of LLM quantization techniques have been proposed, e.g. (Dettmers et al., 2022; Frantar et al., 2023; Chee et al., 2023; Tseng et al., 2024b; Kim et al., 2024; Malinovskii et al., 2025; Lin et al., 2024), exploring the trade-off curve between compression achieved and accuracy loss.

From the accuracy/compression perspective, existing approaches can be clustered into two categories. The first is represented by *lossy compression techniques*, such as Round-to-Nearest (RTN) quantization (Dettmers et al., 2022), llama.cpp (Gerganov & llama.cpp contributors, 2023), GPTQ (Frantar et al., 2023), or AWQ (Lin et al., 2024) which seek to map existing models to popular hardware-supported formats, such as 4-bit grouped weight quantization. These techniques have become extremely popular, and have widespread practical deployment, although they do decrease model accuracy: for example, a recent study by Kurtić et al. (2025) observed that methods such as GPTQ and AWQ can drop zero-shot accuracy on some complex tasks by 3-5% in the case of 4-bit weight quantization. More complex vector-quantized (Egiazarian et al., 2024; Tseng et al., 2024a) or trellis-based (Tseng et al., 2024b) methods can offer reduced accuracy loss at even lower bit-widths, but are hard to apply and interface with state-of-the-art inference pipelines such as vLLM (Kwon et al., 2023), where support for such complex techniques is currently lacking.

The second, more recent, category is given by methods which aim explicitly for *lossless quantization* of the original weights. For instance, ZipNN (Hershcovitch et al., 2025) observes that the exponents of LLM weights are highly compressible, as they do not use the entire bit-width range, reducing average size to approximately 11 bits per parameter (bpp) from 16 bpp, without accuracy loss. Other approaches include entropy-coded FP8 representations (Yang et al., 2025) and dynamic-length BF16 compression (Zhang et al., 2025). On the downside, lossless techniques struggle to reach high compression ratios and require complex kernel support for fast decoding at inference time.

Our Approach. In this paper, we focus on the gap between these two types of approaches, and investigate *Statistically Lossless LLM Quantization (SLQ)*, which we define as compression that may deviate from perfectly matching outputs under deterministic decoding, but maintains the model's output characteristics under standard sampling parameters. Our approach is motivated by the fact that most open-weight models, such as Qwen (Bai et al., 2023) or Llama (Dubey et al., 2024), come with recommended *stochastic* sampling parameters (e.g. temperature $= 0.7$ and $\text{top}_k = 20$ for Qwen3), under which even *unquantized models' outputs vary* by a relative $\sim$1% on zero-shot or user evaluations. The question we ask is: **Is it possible to near-losslessly compress LLMs such that their outputs *statistically* match the original?**

We examine this question along two complementary axes:

- *Task-lossless* (TL) quantization, where zero-shot benchmark accuracy is preserved within sampling variance.
- *Distribution-lossless* (DL) quantization, a stricter criterion where the quantized model's *output distribution* is practically indistinguishable from the original.

Overall, we show that statistically lossless quantization matches realistic deployment scenarios while allowing significant gains beyond lossless compression and inference acceleration. In addition, one key technical finding in the context of SLQ is the role of *asymmetric quantization*, i.e. the presence of a non-trivial zero-point parameter for the quantization grid, as a necessary design choice for achieving Distribution-lossless (DL) fidelity.

- We formalize the distinction between TL and DL compression for LLMs and propose the *expected acceptance rate (EAR)* metric, measuring the maximum token-agreement probability between original and quantized distributions under optimal coupling, as an interpretable output fidelity metric complementary to standard KL-divergence.
- We prove an error gap between symmetric and asymmetric quantization grids. For a weight distribution on $[L, U]$ we define the *centering inefficiency* $\gamma = 2M/R$, with dynamic range $R = U - L$ and $M = \max(|L|, |U|)$. Under standard assumptions (Bennett, 1948), we prove a *gamma-squared variance law* for symmetric grids, saying that they inflate output variance by a factor of γ^2 relative to asymmetric quantization. We are the first to formally connect the asymmetric-vs-symmetric design choice to token-level decision flips, showing that asymmetric quantization greatly improves task-level accuracy.
- We present *SLQ*, a non-uniform scalar quantization pipeline using asymmetric quantization across bitwidths $\{2, 3, 4, 5, 6, 7, 8\}$, with multi-bitwidth Shapley sensitivity estimation and ILP-based allocation. For DL targets, the search operates directly on EAR constraints from calibration data. For TL targets, we introduce a *single-point calibration* procedure that exploits the linear KL-recovery relationship to derive bitwidth requirements from one benchmark measurement. Because allocation follows a fidelity target, SLQ is *Pareto-dominant* over uniform quantizers in both directions: it saves bits where a uniform grid overshoots (TL at **3.59** vs. 4 bpp, Llama-3.3-70B) and spends them where one falls short (QTIP misses TL, Llama-3.1-8B).
- TL compression is achievable as low as **3.3** bpp, while DL requires **5–6 bpp**, corresponding to EAR ≥ 0.99. Humming kernels (InclusionAI, 2025) serve all integer bitwidths efficiently on NVIDIA GPUs, and end-to-end vLLM serving yields 1.7–3.7$\times$ throughput over BF16, consistently outperforming FP8.

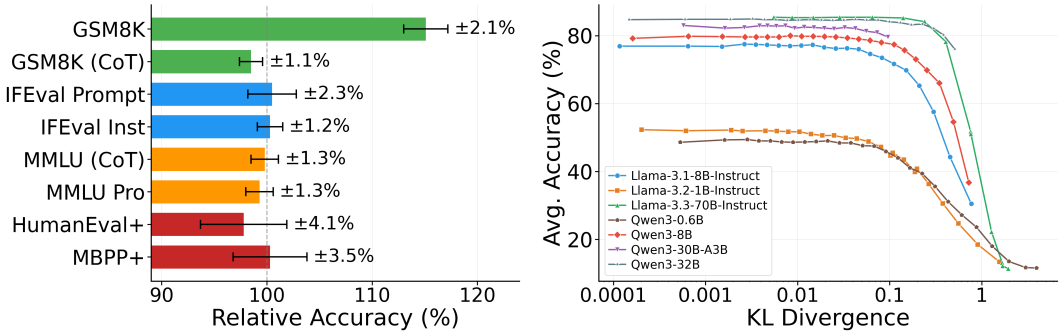
Related Work. Significant prior work focused on *weight-only* post-training quantization, from popular practical tooling (e.g., bitsandbytes (Dettmers et al., 2022)) to algorithmic methods such as GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), SqueezeLLM (Kim et al., 2024), SpQR (Dettmers et al., 2024), OmniQuant (Shao et al., 2024), HQQ (Badri & Shaji, 2023), QuIP and its follow-ups (Chee et al., 2023; Tseng et al., 2024a), additive/vector-quantization approaches (AQLM)(Egiazarian et al., 2024; Van Baalen et al., 2024), as well as more recent trellis-based methods such as QTIP (Tseng et al., 2024b), and YAQA (Tseng et al., 2025). A growing line of work targets *weight-and-activation* quantization via transformations and kernel support, e.g., QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025a). Most techniques target fixed low-bit operating points (e.g., W4 or W4A4), and therefore primarily study the classic compression–accuracy trade-off rather than focusing on the compression achievable in the *near-lossless* regime which is the focus of this paper. Among prior work, only SpQR and YAQA explicitly consider this frontier; nevertheless, SpQR falls well short of the strict accuracy targets we consider, while YAQA relies on substantially heavier system support (e.g., full-model-KL-aware Hessian sketching and integration with specialized quantizers/kernels such as QTIP). In addition, as we show experimentally, YAQA still falls short of the distribution lossless (DL) objective. Complementary to compression methods themselves, Kübler et al. (2026) address how to *decide* whether an optimized model has degraded at all, proposing a McNemar-based hypothesis test that confronts per-sample outcomes rather than aggregated task scores, and thereby separates genuine degradation from evaluation noise. This is close in spirit to our task-lossless criterion: where they provide a test for detecting degradation in a given model, we turn the corresponding fidelity target into the objective that drives the search for the minimum bitwidth.

There has also recently been work on *non-uniform precision allocation* for LLMs: for instance, HIGGS (Malinovskii et al., 2025) leverages linearity-style decompositions to push accuracy at low bitwidths; EvoPress (Sieberling et al., 2025) applies evolutionary search for dynamic compression; OWL applies layer-wise non-uniformity via outlier-guided layer-wise sparsity choices (Yin et al., 2024); and interaction-aware mixed-precision methods (e.g., IMPQ) incorporate cross-layer effects in allocation (Zhao et al., 2025). Our work is complementary: rather than optimizing within the standard lossy region, we specifically focus on the *near-lossless* range where very small distributional shifts matter. We show for the first time that reaching this regime requires (i) asymmetric quantization and (ii) searching across a *wide* set of integer bitwidths rather than a small set of precisions studied by prior work.

A separate line of work targets *lossless* (bit-exact) compression of model weights. ZipNN (Hershcovitch et al., 2025) exploits redundancy in weight representations to achieve lossless compression and efficient (de)compression. Yet, it does not provide inference-time acceleration since weights must be decoded prior to execution: their method is designed to reduce weight storage. Yang et al. (2025) study exponent concentration and propose an entropy-coded *FP8*-centric representation (ECF8) with GPU-optimized decoding, reporting up to 26.9% memory savings and throughput improvements under fixed memory budgets; however, the approach is lossless with respect to an FP8 baseline and therefore applies most directly when models are stored/served in FP8. Finally, DFloat11 (Zhang et al., 2025) introduces dynamic-length, lossless compression of BF16 weights with online GPU decompression for inference under tight memory constraints. These works operate strictly in the lossless regime; for this, they require higher effective bitrates than our work.

2 Motivation

Most open-weight LLMs are deployed with *stochastic decoding*, so even *uncompressed* models exhibit measurable run-to-run variance on zero-shot evaluations (Figure 1, left). A quantized model that stays within this natural variance is, for all practical purposes, lossless. Figure 1b shows that in the near-lossless regime, zero-shot accuracy over standard tasks degrades approximately linearly with the KL divergence between the original and compressed models (Figure 8, Appendix C.5, zooms in on a linear scale). This linearity means a single reference point is sufficient to anchor the compression search, without resorting to arbitrary KL thresholds; we formalize this as a single-point calibration procedure in Section 3.3.



(a) Sampling variance of *uncompressed* Qwen3-8B. (b) KL divergence vs. accuracy (log scale).

Figure 1: (a) Run-to-run variance under stochastic sampling (dotted = greedy). (b) Accuracy vs. KL divergence across seven models; a linear-scale zoom into the near-lossless regime is given in Figure 8, and the comprehensive per-task analysis in Appendix C.5.

Uniform quantization around 4 bits per parameter (bpp) has become the standard target for memory-constrained LLM deployment (Frantar et al., 2023; Lin et al., 2024; Kurtić et al., 2025). However, this target is arbitrary from an accuracy perspective: both GPTQ (Frantar et al., 2023) and QTIP (Tseng et al., 2024b) incur significant KL divergence at $\sim$4 bpp on Llama-3.1-8B-Instruct (Dubey et al., 2024), while our non-uniform allocation achieves KL $= 0.004$ at EAR 0.994 and 5.59 bpp (Table 5, Appendix). More generally, different models and layers have different sensitivities, so any fixed uniform bitwidth is either too aggressive or wasteful. Our approach inverts the problem: given a fidelity target, find the minimum non-uniform bitwidth that achieves it. We validate this across models in Section 4.1.

3 Method

We focus on obtaining near-lossless quantized models via *layer-wise non-uniform scalar quantization*, chosen for its broad support across GPUs (Frantar et al., 2025; 2023; Lin et al., 2024) and CPUs (Gerganov & llama.cpp contributors, 2023; Pegolotti et al., 2023; Ma et al., 2024); the approach should be valid for more complex quantized representations (e.g., vector quantization) as well. Similarly, while we focus on GPTQ as our main quantization method, our approach could work with arbitrary other methods as well. Non-uniform allocation is motivated by the observation that different layers exhibit different sensitivities to quantization (Dong et al., 2019; Kim et al., 2024). Given this target, we first show theoretically that asymmetric quantization is essential for distribution-lossless recovery (Section 3.1). We will then describe the SLQ pipeline: output metrics (Section 3.2), sensitivity estimation (Section 3.3), and bitwidth allocation (Section 3.3).

3.1 Asymmetric vs. Symmetric Quantization

Setup. Consider a weight tensor with values in $[L, U]$. Let $R = U - L$ denote the dynamic range, $M = \max(|L|, |U|)$ the maximum absolute magnitude, and n the number of quantization levels. Symmetric quantization anchors the grid at zero, spanning $[-M, M]$ where $M = \max(|L|, |U|)$. Asymmetric quantization uses a zeropoint to shift the grid, spanning exactly $[L, U]$. Intuitively, symmetric quantization's zero-anchored grid wastes representational capacity when weight distributions are offset from zero, yielding larger per-layer output errors that increase the probability of argmax changes. We provide an overview of the formalization below, showing the impact is *quadratic* in the centering inefficiency. The full argument is in Appendix B.

Definition 3.1 (Centering inefficiency)**.** The centering inefficiency is $\gamma = 2M/R$.

When weights are symmetric about zero, $\gamma = 1$. For asymmetric distributions (typical in LLMs), $\gamma > 1$. For example, weights in $[-0.8, 1.2]$ yield $\gamma = 1.2$: the symmetric grid spans $[-1.2, 1.2]$, wasting 20% of its capacity on the interval $[-1.2, -0.8]$ where no weights exist.

From step sizes to noise variance. Symmetric quantization has step size $\Delta_{\text{sym}} = 2M/(n-1)$, while asymmetric has $\Delta_{\text{asym}} = R/(n-1)$. Their ratio is $\Delta_{\text{sym}}/\Delta_{\text{asym}} = 2M/R = \gamma$, independent of the number of levels. Under the high-rate approximation (Bennett, 1948), quantization error is approximately uniform within each bin, with variance $\sigma^2 = \Delta^2/12$. Since $\Delta_{\text{sym}} = \gamma \cdot \Delta_{\text{asym}}$, the variance inherits a *quadratic* dependence:

Lemma 3.2 (Gamma-Squared Variance Law)**.** *The quantization noise variance under symmetric quantization scales quadratically with the centering inefficiency relative to asymmetric quantization:*

$$\sigma^2_{\text{sym}} = \gamma^2 \, \sigma^2_{\text{asym}}. \tag{1}$$

Impact on output error. For a single linear layer $\mathbf{y} = W\mathbf{x}$, quantization produces output perturbation $\boldsymbol{\delta} = (Q(W) - W)\mathbf{x}$, with expected squared norm $\sigma^2\|\mathbf{x}\|^2$. Since quantization is applied to identical inputs, $\|\mathbf{x}\|^2$ cancels out:

$$\frac{\mathbb{E}[\|\boldsymbol{\delta}_{\text{sym}}\|^2]}{\mathbb{E}[\|\boldsymbol{\delta}_{\text{asym}}\|^2]} = \gamma^2. \tag{2}$$

This implies that a distribution with $\gamma = 1.2$, which is only mildly asymmetric, suffers 44% more per-layer output error under symmetric quantization. This compounds across layers and increases the probability of *decision flips*: positions where quantization perturbs logits enough to change the argmax token, directly reducing EAR. The full derivation is in Appendix B.

Empirical Validation. We now validate the γ^2 scaling on Llama-3.1-8B-Instruct (uniform 4-bit). Figures 2–3 show symmetric quantization flips more tokens across all entropy bins, with the ratio largest at low entropy (up to 3×). The resulting EAR gap is clearest at the DL threshold itself: under Shapley allocation over the full bitwidth range, asymmetric quantization reaches EAR $\geq$ 0.99 at 5.5 bpp, whereas symmetric does not reach it until 7.0 bpp, a gap of 1.5 bpp (Appendix G.1). Figure 4 confirms that the effect compounds across layers: symmetric quantization degrades more sharply in the first few sensitive layers, undermining mixed-precision schemes. On MMLU-Pro, where each answer is a single-token decision, asymmetric quantization recovers 88.4% of FP16 accuracy versus 82.5% for symmetric (Appendix C.2).

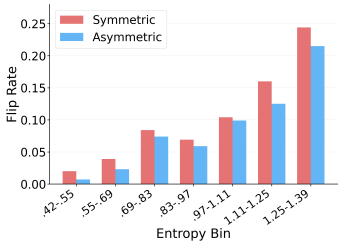


Figure 2: Decision flip rates.

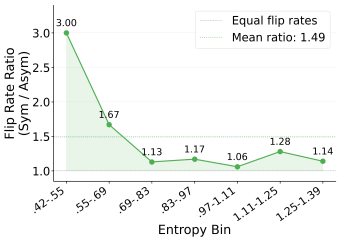


Figure 3: Sym-to-asym ratio.

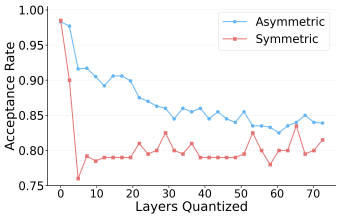


Figure 4: Layer-by-layer.

These experiments show that the asymmetric–symmetric gap is most pronounced at the distributional level: achieving EAR $\geq$ 0.99 with symmetric quantization requires up to 1.5 additional bits per parameter. For task-level accuracy, symmetric quantization preserves high average recovery, because stochastic sampling absorbs the additional token flips without measurably affecting aggregate scores. We conclude that asymmetric quantization is therefore a prerequisite for distribution-lossless compression, but symmetric remains viable when only task-level preservation is required.

Relation to prior work. That asymmetric grids reduce reconstruction error is known. The consequence at the token level has not been made explicit. The γ^2 inflation causes symmetric quantization to flip tokens that asymmetric leaves intact, and the flip-rate ratio is widest in the low-entropy bins, where the original model was most confident (Figure 3). The grid geometry therefore determines whether a given bit budget can reach the DL threshold.

3.2 Output Metrics for Losslessness

The non-uniform quantization problem is that of assigning a bitwidth $b_\ell \in \mathcal{B} = \{2, 3, 4, 5, 6, 7, 8\}$ to each layer ℓ. The average bitwidth is $\bar{b}(\mathbf{b}) = \sum_\ell b_\ell |W_\ell| / \sum_\ell |W_\ell|$. The first challenge is that of precise output metrics, computable from a small calibration set, that reliably predict when a configuration is near-lossless.

KL-Divergence. For N calibration token positions with original and quantized distributions p_i, q_i, we compute $D_{\mathrm{KL}} = \frac{1}{N} \sum_i \sum_{k \in \mathcal{T}_i} p_i(k) \log \frac{p_i(k)}{q_i(k)}$, restricted to the top-$K$=10 tokens under p_i. However, KL divergence is measured in natural log units (nats), unbounded above, and asymmetric. Its magnitude is difficult to interpret without calibrated intuition for a given vocabulary and distribution shape. While Pinsker's inequality ($d_{TV} \leq \sqrt{D_{\mathrm{KL}}/2}$) bounds KL into a percentage-like quantity, this bound is loose.

Expected Acceptance Rate (EAR). We therefore define the *Expected Acceptance Rate (EAR)* as the primary metric for *distribution-lossless* compression, inspired by the token acceptance probability in speculative decoding (Leviathan et al., 2023). In that setting, a draft token sampled from a proposal distribution q is accepted under the target distribution p with probability $\sum_k \min(p(k), q(k))$; the same quantity measures the fidelity of a quantized model's output distribution to the original. Formally, this probability mass overlap equals $1 - d_{TV}(p, q)$, is bounded in $[0, 1]$, and is directly interpretable as a percentage of shared probability mass. By the coupling characterization of total variation distance, $\sum_k \min(p(k), q(k))$ equals the maximum probability that two random variables $X \sim p$ and $Y \sim q$ can be coupled to agree ($X = Y$). We average this over N calibration positions:

$$\mathrm{EAR} = \frac{1}{N} \sum_i \sum_{k=1}^{K} \min(p_i(k), q_i(k)), \tag{3}$$

where both distributions are restricted to the top-K tokens under p_i. EAR has a direct operational interpretation: EAR $= 0.99$ means that, under optimal coupling, the original and quantized models produce the same token 99% of the time. While the underlying quantity $1 - d_{TV}$ is classical, its application as a compression fidelity criterion for quantized LLMs is, to our knowledge, new.

Restricting to top-K tokens is a tight approximation: LLM output distributions are sharply peaked, so the excluded tail contributes negligible mass. EAR is typically more restrictive than KL (EAR ≥ 0.99 generally implies $D_{\mathrm{KL}} \ll 0.01$), and is robust to tail behavior since $\min(p, q)$ clips near-zero probabilities rather than producing unbounded penalties. We use EAR as the primary constraint for bitwidth search (Section 3.3), with KL as a secondary diagnostic. Both are computed simultaneously during each forward pass at no additional cost.

3.3 Sensitivity Estimation and Bitwidth Allocation

Given the output metrics, we need to assign each layer a bitwidth from the budget $\mathcal{B} = \{2, 3, 4, 5, 6, 7, 8\}$. Prior mixed-precision work typically uses binary allocation (e.g., a layer is set to either 4 or 8 bits) (Zhao et al., 2025; Dong et al., 2019). We find this experimentally to be too coarse for the near-lossless regime: restricting to $\{4, 8\}$ consistently requires higher average bitwidth than the full range to meet the same quality constraint (see Section 4.1). Since inference engines often require certain layers to share a bitwidth (e.g., fused QKV projections in vLLM), we partition layers into M groups $\mathcal{G} = \{G_1, \ldots, G_M\}$.

Sensitivity Estimation. We need to estimate how each group's quantization at a given bitwidth affects the output metrics. A simple baseline is *linear estimation* (Malinovskii et al., 2025): model metric degradation as $\Delta_{\mathrm{KL}}(\mathbf{b}) \approx \sum_m \alpha_m \cdot e_m^{(b_m)}$, where $e_m^{(b)}$ is the normalized reconstruction error and α_m is estimated via noise injection ($O(T \cdot M)$ forward passes). This is fast, but assumes independence across groups.

Multi-Bitwidth Shapley Estimation. Shapley-based sensitivity estimation (Shapley, 1953) has been applied to mixed-precision quantization (Zhao et al., 2025),

pruning (Adamczewski et al., 2024; Sun et al., 2025), and layer importance analysis (Zhang et al., 2024). However, existing approaches consider layer-wise binary choices, switching layers between one high and one low precision, which restricts allocation to a binary choice and would thus lead to higher average bitwidths (Section 4.1 and Appendix G.1).

We extend this to support the full bitwidth range by running a *separate* binary game for each target bitwidth $b^* \in \mathcal{B} \setminus \{b_{\max}\}$. In each game, we sample P random permutations, start with all groups at $b_{\max}$, and sequentially switch each group to b^*, recording the marginal metric change. The Shapley value $\phi_m^{(b^*)}$ for group G_m is the average marginal contribution across permutations. Across all $|\mathcal{B}| - 1$ games, this yields a full sensitivity database $\phi_m^{(b)}$ for every group-bitwidth pair. The games are independent and can run in parallel; each forward pass computes KL and EAR simultaneously. The total cost is $O(P \cdot M \cdot |\mathcal{B}|)$ forward passes. The full procedure is given in Algorithm 1 (Appendix A).

Bitwidth Allocation. Given the sensitivity database $c_{m,b}$ from either estimation method, we solve a multiple-choice knapsack problem via integer linear programming (ILP): assign each group G_m a bitwidth b to minimize total predicted degradation $\sum_{m,b} x_{m,b} \cdot c_{m,b}$ subject to a bitwidth budget. We then binary-search over the budget to find the minimum average bitwidth satisfying a quality constraint. The two compression targets use different constraints, described below. The same framework extends to joint weight-and-activation quantization by quantizing activations in the sensitivity estimates and allocation (Appendix G.1). The sensitivity database is computed once, and each ILP solve then takes milliseconds, so the binary search adds little cost. SLQ is in principle solver-agnostic: more accurate alternatives such as gradient-based search (Helcig & Alistarh, 2026) are slower, but remain fast enough to be practical. Details on metric prediction and an alternative evolutionary search appear in Appendices A.2 and A.4. Appendix C.1 visualizes the per-layer sensitivity landscape and resulting bitwidth assignments.

Distribution-lossless search. The sensitivity database directly predicts EAR and KL for any candidate configuration (Appendix A.2). We set a target such as EAR $\geq$ 0.99 and binary-search over the bitwidth budget until the predicted EAR meets the constraint. The entire search operates on calibration-set metrics and requires no downstream benchmark evaluation, and is stable across calibration seeds: repeated runs agree to within ± 0.0005 EAR.

Task-lossless search via single-point calibration. For task-lossless compression, the goal is to find the minimum bitwidth that preserves a target recovery rate (e.g., $\geq 99\%$ of BF16 accuracy) across all benchmarks. We exploit the approximate linearity between KL divergence and task recovery established in Section 2 (Figure 1b):

$$\text{recovery} \approx 1 - \alpha \cdot D_{\text{KL}}. \tag{4}$$

This linear regime holds reliably for the high-bitwidth configurations relevant to near-lossless compression, where the perturbation is small enough that accuracy degrades smoothly with distributional shift. We observe this consistently across six model families and five task categories (Appendix C.5): within the near-lossless range, recovery is well-approximated by a single slope, while at more aggressive compression the relationship becomes non-linear and task-dependent.

The intercept is known (D_{KL}=0 implies recovery=1), so a single calibration point determines the slope α. We choose a uniform bitwidth where the model shows meaningful degradation (e.g., uniform 4-bit), measure its actual KL divergence on calibration data, and evaluate benchmarks to obtain recovery. This yields $\alpha = (1 - \text{recovery})/D_{\text{KL}}^{\text{cal}}$, and inverting gives the KL threshold for any target recovery: $D_{\text{KL}}^{\text{thresh}} = (1 - \text{target recovery})/\alpha$.

Since Shapley-predicted KL may differ from measured KL by a systematic factor, we compute a calibration ratio $\rho = D_{\text{KL}}^{\text{actual}}/D_{\text{KL}}^{\text{predicted}}$ at the calibration point and apply it to all subsequent predictions: $\hat{D}_{\text{KL}} = \rho \cdot D_{\text{KL}}^{\text{predicted}}$. The binary search then checks $\hat{D}_{\text{KL}} \leq D_{\text{KL}}^{\text{thresh}}$ at each candidate bitwidth, requiring no additional forward passes. The total cost is one forward pass to measure KL plus two benchmark runs (BF16 baseline and calibration

configuration); the search itself is arithmetic on the sensitivity database. The full procedure is given in Algorithm 2 (Appendix A.3).

Guardrail against out-of-regime anchors. Since Equation 4 and ρ are fitted from a single point, an anchor outside the near-lossless regime would extrapolate a poor threshold. We therefore add a held-out check: once the search converges we measure the actual KL and reject the configuration if the actual-to-predicted ratio diverges from ρ by more than $2\times$, re-anchoring elsewhere. This costs one forward pass; within the regime α is nearly constant, so 4-bit and 6-bit anchors agree, while 3-bit is not recommended.

4 Experiments

We evaluate SLQ for weight-only quantization on four models: Qwen3-8B, Qwen3-32B, Qwen3.5-27B (Bai et al., 2023), and Llama-3.3-70B-Instruct (Dubey et al., 2024), and on the Qwen3.5-35B-A3B mixture-of-experts model (Appendix F). We quantize layers using the standard GPTQ technique. Table 1 reports per-benchmark scores and average recovery rate (Avg Rec.) relative to the BF16 baseline under non-greedy decoding, averaged over 3 seeds; values of 1.0 indicate exact baseline recovery. Each model block shows SLQ-DL and SLQ-TL configurations alongside uniform W4A16 or GPTQ-Int4 references. We report average bits per parameter, accounting for all quantization overheads (Appendix F).

Setup. Following our prior analysis, all models use asymmetric integer group quantization (group size 128, bitwidths $\{2, 3, 4, 5, 6, 7, 8\}$) with Shapley-based per-layer allocation (Section 3.3) using 512 calibration samples; DL targets use EAR $\geq$ 0.985–0.99 depending on the model. The uniform W4A16 baselines use GPTQ (Frantar et al., 2023) with tuned hyperparameters; SLQ is orthogonal to the choice of per-layer quantizer and instead provides a principled method for finding the minimum-bitwidth non-uniform configuration that meets a given fidelity target. Quantized weights are served with Humming kernels (InclusionAI, 2025) in vLLM (Kwon et al., 2023) (MARLIN-based (Frantar et al., 2025), 4–8 bit; Appendix C.4). Zero-shot accuracy is measured on seven benchmarks: reasoning (ARC-Challenge, GSM8K, GSM8K-CoT), instruction following (IFEval), and knowledge (MMLU-CoT, MMLU-Pro), under non-greedy decoding with model-recommended sampling parameters, averaged over 3 seeds. End-to-end throughput is measured on NVIDIA L40s GPUs, running the Humming kernels in vLLM, using ShareGPT and Reasoning workloads; we report output tokens per second per GPU (TPS/GPU) to normalize across tensor-parallelism configurations.

4.1 Accuracy Results

Zero-shot accuracy as a fidelity metric. In the near-lossless regime, zero-shot benchmarks are a blunt instrument: models with meaningfully different distributional fidelity can score within each other's noise bands (Figure 1a). This is why we use EAR/KL as the primary search objective for DL and a recovery threshold for TL, rather than optimizing benchmark scores directly.

Distribution-lossless results. The SLQ-DL rows show configurations optimized for distributional fidelity (EAR $\geq$ 0.99). All four models achieve $\geq$99.3% average recovery at 4.98–6.55 bits, within the sampling variance of the baseline on every benchmark. Qwen3-32B compresses most aggressively (4.98 bits, 100.1% recovery), while Llama-3.3-70B requires 6.55 bits.

Task-lossless at aggressive bitwidths. When the goal is only task-level preservation, the single-point calibration procedure (Section 3.3) allows substantially more aggressive bitwidths. The SLQ-TL configurations target $\geq$99% average recovery and achieve it across all four models: Qwen3.5-27B at 3.30 bits, Llama-3.3-70B at 3.59 bits, Qwen3-8B at 4.72 bits, and Qwen3-32B at 3.59 bits. In every case, SLQ-TL dominates uniform W4A16 at comparable or lower bitwidths.

[1]Bits averaged over subset (MLP + linear attention input/output projections); remaining layers are BF16. Following Qwen's convention; Qwen3.5-27B-GPTQ-Int4 labeled *Int4* under the same scheme.

Table 1: Non-greedy accuracy and recovery rates. W4A16 shown for reference. Bold only for visibility. GPTQ-Int4 is the official 4-bit quantized Qwen3.5-27B. [1]

Model	Bits	ARC Chal.	GSM8K	GSM8K (CoT)	IFEval Prompt	IFEval Inst.	MMLU CoT	MMLU Pro	Avg Rec.
Qwen3-8B (non-greedy, temp=0.6, top_p=0.95, top_k=20, 3 seeds)									
Baseline (BF16)	16.0	90.64	79.15	90.75	82.44	88.13	71.23	63.42	1.000
SLQ-DL	5.70	90.30	80.44	90.45	80.96	87.17	68.95	63.45	0.993
SLQ-TL	4.72	90.13	80.43	90.60	82.99	88.49	71.86	62.92	1.003
Uniform-4	4.16	89.27	80.74	90.22	82.26	87.77	60.01	61.80	0.976
Qwen3-32B (non-greedy, temp=0.6, top_p=0.95, top_k=20, 3 seeds)									
Baseline (BF16)	16.0	92.50	91.72	91.73	86.40	89.22	86.30	70.70	1.000
SLQ-DL	4.98	92.28	93.27	91.48	86.53	89.82	87.32	70.12	1.003
SLQ-TL	3.59	92.30	93.33	91.22	85.89	88.90	86.01	68.92	0.996
Uniform-4	4.16	91.88	91.39	90.90	86.01	89.12	85.44	68.90	0.991
Qwen3.5-27B (non-greedy, temp=1.0, top_p=0.95, top_k=20, 3 seeds)									
Baseline (BF16)	16.0	96.88	96.84	95.73	86.08	90.49	88.16	77.87	1.000
SLQ-DL	5.19	95.28	96.78	95.45	85.77	90.17	88.47	78.43	0.997
SLQ-TL	3.30	96.22	95.75	94.84	85.21	89.81	87.53	76.64	0.991
GPTQ-Int4	4.1	96.65	96.66	95.56	86.41	90.65	88.23	77.68	0.999
Llama-3.3-70B-Instruct (non-greedy, temp=0.6, top_p=0.9, 3 seeds)									
Baseline (BF16)	16.0	93.45	93.38	94.52	88.78	92.37	86.39	70.78	1.000
SLQ-DL	6.55	93.48	93.55	94.80	89.52	92.85	86.26	70.52	0.999
SLQ-TL	3.59	92.70	93.33	93.48	89.28	92.69	85.19	68.25	0.991
Uniform-4	4.16	93.65	91.10	93.71	89.34	92.57	85.94	70.08	0.998

Table 2: QTIP and YAQA evaluated by us on the released checkpoints, under the benchmark suite and sampling setup of Table 1. Task-losslessness requires $\geq 99\%$ mean recovery, distribution-losslessness EAR ≥ 0.99; no configuration reaches the latter.

Method	Model	Bits	Task mean rec.	KL $\downarrow$	EAR $\uparrow$
QTIP	Llama-3.1-8B	4-bit	97.94%	0.0172	0.9519
QTIP	Llama-3.1-8B	3-bit	96.00%	0.0510	0.9165
QTIP	Llama-3.3-70B	4-bit	99.9%	0.0244	0.9544
YAQA	Llama-3.3-70B	4-bit	99.5%	0.0150	0.9622

Comparison to state-of-the-art quantizers. QTIP (Tseng et al., 2024b) and YAQA (Tseng et al., 2025) are orthogonal to SLQ, so the question is whether uniform allocation with a stronger quantizer already reaches the statistically-lossless regime. Table 2 shows it does not. On Llama-3.1-8B, QTIP misses both targets at every bitwidth tested. On Llama-3.3-70B, QTIP and YAQA at 4 bits clear the task-lossless bar, but their EAR leaves them far from the 0.99 required for distribution-losslessness. SLQ reaches TL on that model at 3.59 bpp and DL at 6.55 bpp (Table 1), and on Llama-3.1-8B attains KL $= 0.004$ at 5.59 bpp against QTIP's 0.017 at 4 bits. A uniform budget is set without reference to any fidelity target, so it lands on the wrong side of the boundary in both directions.

Comparison to non-uniform allocators. HIGGS (Malinovskii et al., 2025), EvoPress (Sieberling et al., 2025) and IMPQ (Zhao et al., 2025) maximize quality under a fixed budget, and so cannot *locate* the losslessness threshold without sweeping the entire curve. Against IMPQ on Qwen3-8B, task recovery saturates for both above $\sim$4 bpp while EAR separates them (Appendix D.1).

Ablations. The full bitwidth range $\{4, 5, 6, 7, 8\}$ is essential: on Qwen3-32B at EAR ≥ 0.99, restricting to $\{4, 8\}$ requires 5.74 bits while the full range achieves equivalent fidelity at 4.98 bits. Shapley-based allocation outperforms linear estimation at mid-range bitwidths

(5.5–6.5 bits) where decisions are tightest. Comprehensive ablations and bitwidth-range comparisons appear in Appendix G.1.

Token-Level Analysis for DL. We perform a teacher-forced token-level comparison between base Qwen3-8B and its 5.70-bit SLQ-DL quantization: responses generated by the base model (greedy) are fed through the quantized model to record per-position top-1 disagreements, isolating quantization effects from autoregressive divergence. Across 15 diverse prompts (6,531 tokens), 98.1% of tokens are identical; the 1.9% that diverge are minor stylistic variations (synonym swaps, punctuation) with no factual errors, distributed uniformly across positions. Per-prompt statistics and annotated examples are in Appendix F.1.

Thinking token inflation. Thinking tokens dominate reasoning model inference cost (Chen et al., 2026). On Qwen3-8B GSM8K (1319 problems, 3 seeds), SLQ-DL introduces *no* additional thinking overhead relative to BF16, whereas SLQ-TL and Uniform-4 increase thinking tokens by +2.2% and +2.9%, respectively (see Appendix G for full details). The gap is statistically clean (paired Wilcoxon: $p = 0.86$ for SLQ-DL, $p < 10^{-5}$ for both lossy configurations), so TL and DL are separate regimes rather than points on one axis: a model can be task-lossless and still cost measurably more to serve.

4.2 Inference Speedup

Table 3 presents end-to-end throughput on L40S GPUs, running the Humming kernels (InclusionAI, 2025) in vLLM. The SLQ configurations provide 1.7–3.7× throughput over BF16 at $\geq$ 99.1% recovery, consistently outperforming FP8 per GPU (1.23× on Qwen3-8B, 1.54× on Qwen3-32B). Speedups come from both faster kernel execution and reduced GPU count: Qwen3-32B fits on 1 GPU instead of 2, Llama-3.3-70B on 2 instead of 4.

Table 3: End-to-end throughput on L40s GPUs (Humming kernels in vLLM). TPS/GPU normalizes by GPU count.

Model	Bits	GPUs	ShareGPT		Reasoning	
			TPS/GPU	Speedup	TPS/GPU	Speedup
Qwen3-8B						
BF16	16.0	1	630.9	1.00×	544.6	1.00×
SLQ-DL	**5.70**	**1**	**1097.2**	**1.74×**	**907.6**	**1.67×**
SLQ-TL	**4.72**	**1**	**1223.9**	**1.94×**	**1010.9**	**1.86×**
FP8	8.0	1	894.3	1.42×	748.4	1.37×
Qwen3-32B						
BF16	16.0	2	148.1	1.00×	135.5	1.00×
SLQ-DL	**4.98**	**1**	**435.5**	**2.94×**	**383.5**	**2.83×**
FP8	8.0	1	282.8	1.91×	121.6	0.90×
Llama-3.3-70B-Instruct						
BF16	16.0	4	62.8	1.00×	60.8	1.00×
SLQ-DL	**6.55**	**2**	**139.8**	**2.23×**	**129.7**	**2.13×**
SLQ-TL	**3.59**	**2**	**231.1**	**3.68×**	**190.9**	**3.14×**
FP8	8.0	4	93.8	1.50×	88.2	1.45×

5 Conclusion & Future Work

We formalized statistically-lossless LLM compression, distinguishing task-lossless (TL) and distribution-lossless (DL) targets, and proposed EAR as a fidelity metric. The γ^2 variance law shows asymmetric quantization is a prerequisite for DL. SLQ achieves TL at 3.3–4.7 bits, DL at 5.0–6.6 bits, and 1.7–3.7× speedup over BF16 across four models, and transfers unchanged to a mixture-of-experts model. Limitations are discussed in Appendix E; future work includes weight-and-activation quantization and speculative decoding.

Ethics Statement

This paper presents research aimed at advancing the field of Machine Learning. We do not foresee any specific ethical concerns arising directly from this work that warrant particular emphasis beyond the broader societal implications common to the field.

Acknowledgments

We acknowledge the use of Humming kernels developed by Jinzhen Lin and the Venus Team, Ant Group.

References

Kamil Adamczewski, Yawei Li, and Luc Van Gool. Shapley pruning for neural network compression. *arXiv preprint arXiv:2407.15875*, 2024.

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 *Advances in Neural Information Processing Systems*, volume 37, 2024.

Hicham Badri and Appu Shaji. HQQ: Half-quadratic quantization of large machine learning models. Technical Blog Post, Mobius Labs, 2023. URL `https://dropbox.github.io/hqq_blog/`.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

William Ralph Bennett. Spectra of quantized signals. *The Bell System Technical Journal*, 27(3): 446–472, 1948.

Julian J. Bussgang. Crosscorrelation functions of amplitude-distorted gaussian signals. *Research Laboratory of Electronics, Massachusetts Institute of Technology, Technical Report 216*, 1952.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Lingjiao Chen, Chi Zhang, Yeye He, Ion Stoica, Matei Zaharia, and James Zou. The price reversal phenomenon: When cheaper reasoning models cost more. *arXiv preprint arXiv:2603.23971*, 2026.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 239–251, 2025.

Georgi Gerganov and llama.cpp contributors. llama.cpp: LLM inference in C/C++. `https://github.com/ggml-org/llama.cpp`, 2023. Accessed: 2026-01-28.

Michael Helcig and Dan Alistarh. Model compression with exact budget constraints via riemannian manifolds. *arXiv preprint arXiv:2605.00649*, 2026.

Moshik Hershcovitch, Andrew Wood, Leshem Choshen, Guy Girmonsky, Roy Leibovitz, Or Ozeri, Ilias Ennmouri, Michal Malka, Peter Chin, Swaminathan Sundararaman, and Danny Harnik. ZipNN: Lossless compression for AI models. In *2025 IEEE 18th International Conference on Cloud Computing (CLOUD)*, pp. 186–198. IEEE, 2025.

InclusionAI. Humming: A JIT-compiled GEMM kernel library for quantized inference, 2025. URL `https://github.com/inclusionAI/humming`. Open-source JIT-compiled GEMM kernel library for quantized inference, supporting symmetric integer weight formats from 1 to 8 bits.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR. PMLR, 2024.

Jonas Kübler, Kailash Budhathoki, Matthäus Kleindessner, Xiong Zhou, Junming Yin, Ashish Khetan, and George Karypis. When LLMs get significantly worse: A statistical approach to detect model degradations. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

Eldar Kurtić, Alexandre Noll Marques, Shubhra Pandit, Mark Kurtz, and Dan Alistarh. "give me BF16 or give me death"? accuracy-performance trade-offs in LLM quantization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26872–26886, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23. ACM, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 19274–19286. PMLR, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. Best Paper Award.

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 *Proceedings of the 13th International Conference on Learning Representations*, 2025a.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. ParetoQ: Improving scaling laws in extremely low-bit LLM quantization. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

S. Ma et al. The era of 1-bit LLMs: All large language models are in 1.58 bits, 2024.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. HIGGS: Pushing the limits of large language model quantization via the linearity theorem. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10857–10886, 2025.

Tommaso Pegolotti, Elias Frantar, Dan Alistarh, and Markus Püschel. Qigen: Generating efficient kernels for quantized inference on large language models. *arXiv preprint arXiv:2307.03738*, 2023.

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 *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, 1953.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtić, and Dan Alistarh. EvoPress: Accurate dynamic model compression via evolutionary search. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Chuan Sun, Han Yu, Lizhen Cui, and Xiaoxiao Li. Efficient Shapley value-based non-uniform pruning of large language models. *arXiv preprint arXiv:2505.01731*, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR. PMLR, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. Spotlight.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. *arXiv preprint arXiv:2505.22988*, 2025. To appear at ICML 2026.

Mart Van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*, 2024.

Zeyu Yang, Tianyi Zhang, Jianwen Xie, Chuan Li, Zhaozhuo Xu, and Anshumali Shrivastava. To compress or not? pushing the frontier of lossless GenAI model weights compression with exponent concentration. *arXiv preprint arXiv:2510.02676*, 2025.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Tianyi Zhang, Mohsen Hariri, Shaochen Zhong, Vipin Chaudhary, Yang Sui, Xia Hu, and Anshumali Shrivastava. 70% size, 100% accuracy: Lossless LLM compression for efficient GPU inference via dynamic-length float (DFloat11). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=xdNAVP7TGy`.

Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 469–479. Association for Computational Linguistics, 2024.

Junchen Zhao, Ali Derakhshan, Dushyant Bharadwaj, Jayden Kana Hyman, Junhao Dong, Sangeetha Abdu Jyothi, and Ian Harris. IMPQ: Interaction-aware layerwise mixed precision quantization for LLMs. *arXiv preprint arXiv:2509.15455v1*, 2025.

Appendix

A Algorithms and Metric Prediction

A.1 Multi-Bitwidth Shapley Estimation

Algorithm 1 details the multi-bitwidth Shapley estimation procedure described in Section 3.3.

Algorithm 1 Multi-Bitwidth Shapley Estimation

Require: Groups $\mathcal{G} = \{G_1, \ldots, G_M\}$, bitwidths $\mathcal{B}$, permutations P
Ensure: Cost table $\phi_m^{(b)}$ for all groups m, bitwidths b

```
for each target bitwidth b* ∈ B \ {b_max} do
    for p = 1, ..., P do
        Sample random permutation π of {1, ..., M}
        Set all groups to b_max; evaluate metrics
        for j = 1, ..., M do
            Switch group G_{π_j} from b_max to b*
            Evaluate metrics (KL, EAR)
            Record marginal change Δ_{π_j}^{(b*)}
        end for
    end for
    φ_m^{(b*)} ← (1/P) Σ_p Δ_m^{(b*)} for each group m
end for
```

A.2 Metric Prediction from Sensitivity Data

During binary search, the predicted evaluation mode estimates KL divergence and EAR from sensitivity coefficients without running model inference. This enables rapid iteration over candidate bitwidth budgets.

Linear Method. Under the independence assumption, total metric degradation is the sum of per-group contributions:

$$\widehat{\Delta D_{\mathrm{KL}}}(\mathbf{b}) = \sum_m e_m^{(b_m)} \cdot \alpha_m^{\mathrm{KL}} \tag{5}$$

$$\widehat{\Delta \mathrm{EAR}}(\mathbf{b}) = \sum_m e_m^{(b_m)} \cdot \alpha_m^{\mathrm{EAR}} \tag{6}$$

where $e_m^{(b)}$ is the normalized reconstruction error for group G_m at bitwidth b, computed as $e_m^{(b)} = \frac{1}{|G_m|} \sum_{\ell \in G_m} \|W_\ell - \hat{W}_\ell^{(b)}\|_F^2 / \|W_\ell\|_F^2$. The coefficients α_m^{KL} and α_m^{EAR} are estimated via noise injection as described in Section 3.3. The predicted metrics are then $\widehat{D_{\mathrm{KL}}} = \widehat{\Delta D_{\mathrm{KL}}}$ and $\widehat{\mathrm{EAR}} = 1 - \widehat{\Delta \mathrm{EAR}}$.

Shapley Method. The Shapley values $\phi_m^{(b)}$ directly represent expected metric degradation from assigning group m to bitwidth b relative to the reference bitwidth $b_{\max}$:

$$\widehat{D_{\mathrm{KL}}}(\mathbf{b}) = \sum_m \phi_m^{\mathrm{KL},(b_m)} \tag{7}$$

$$\widehat{\mathrm{EAR}}(\mathbf{b}) = 1 - \sum_m \phi_m^{\mathrm{EAR},(b_m)} \tag{8}$$

Since Shapley values are computed using actual quantized weights during permutation sampling, they capture both reconstruction error magnitude and its propagation through the network. For unified permutation sampling, which outputs transition costs rather than absolute costs, we apply cumulative summation from $b_{\max}$ downward before use in the solver.

A.3 Task-Lossless Single-Point Calibration

Algorithm 2 details the single-point calibration procedure for task-lossless bitwidth search described in Section 3.3.

Algorithm 2 Task-Lossless Single-Point Calibration Search

Require: Sensitivity database $\phi_m^{(b)}$, ILP solver, target recovery r^*, calibration bitwidth b_{cal}
Ensure: Minimum-bitwidth configuration $\mathbf{b}^*$ with recovery $\geq r^*$

1: **// Step 1: Calibrate**
2: $\mathbf{b}_{\mathrm{cal}} \leftarrow$ ILP-SOLVE$(\phi, \bar{b} = b_{\mathrm{cal}})$ {Proxy-optimal config at b_{cal}}
3: $D_{\mathrm{KL}}^{\mathrm{actual}} \leftarrow$ MEASUREKL$(\mathbf{b}_{\mathrm{cal}})$ {One forward pass}
4: $D_{\mathrm{KL}}^{\mathrm{predicted}} \leftarrow \sum_m \phi_m^{(b_{\mathrm{cal},m})}$ {Shapley prediction}
5: $\rho \leftarrow D_{\mathrm{KL}}^{\mathrm{actual}} / D_{\mathrm{KL}}^{\mathrm{predicted}}$ {Calibration ratio}
6: $\mathrm{recovery}_{\mathrm{cal}} \leftarrow$ BENCHMARK$(\mathbf{b}_{\mathrm{cal}})$ / BENCHMARK(BF16)
7: **// Step 2: Fit linear model**
8: $\alpha \leftarrow (1 - \mathrm{recovery}_{\mathrm{cal}}) \,/\, D_{\mathrm{KL}}^{\mathrm{actual}}$ {Slope from single point}
9: $D_{\mathrm{KL}}^{\mathrm{thresh}} \leftarrow (1 - r^*) \,/\, \alpha$ {KL threshold for target recovery}
10: **// Step 3: Binary search (no forward passes)**
11: $b_{\mathrm{lo}} \leftarrow \min(\mathcal{B})$, $b_{\mathrm{hi}} \leftarrow \max(\mathcal{B})$
12: **while** $b_{\mathrm{hi}} - b_{\mathrm{lo}} > \epsilon$ **do**
13: $\quad b_{\mathrm{mid}} \leftarrow (b_{\mathrm{lo}} + b_{\mathrm{hi}})/2$
14: $\quad \mathbf{b} \leftarrow$ ILP-SOLVE$(\phi, \bar{b} = b_{\mathrm{mid}})$
15: $\quad \hat{D}_{\mathrm{KL}} \leftarrow \rho \cdot \sum_m \phi_m^{(b_m)}$ {Calibrated prediction}
16: $\quad$ **if** $\hat{D}_{\mathrm{KL}} \leq D_{\mathrm{KL}}^{\mathrm{thresh}}$ **then**
17: $\quad\quad b_{\mathrm{hi}} \leftarrow b_{\mathrm{mid}}$; $\mathbf{b}^* \leftarrow \mathbf{b}$
18: $\quad$ **else**
19: $\quad\quad b_{\mathrm{lo}} \leftarrow b_{\mathrm{mid}}$
20: $\quad$ **end if**
21: **end while**
22: **Return:** $\mathbf{b}^*$

The calibration step (lines 2–6) requires one forward pass and two benchmark runs. The binary search (lines 10–19) uses only the sensitivity database and ILP solver, with no additional model evaluations.

A.4 Evolutionary Search as an Alternative Solver

As an alternative to the ILP-based solver (Section 3.3), bitwidth allocation can be formulated as a combinatorial search problem. We implement a variant of EvoPress (Sieberling et al., 2025) adapted to our constraint-based setting: whereas the original minimizes loss subject to a bitwidth budget, we minimize bitwidth subject to a quality constraint. Algorithm 3 summarizes the procedure.

Algorithm 3 Constraint-Based Evolutionary Bitwidth Search

Input: Groups $\mathcal{G}$, quality threshold τ, offspring count λ
Initialize $\mathbf{b}^{(0)} \leftarrow$ uniform at $\max(\mathcal{B})$, `stall` $\leftarrow 0$
while not converged **do**
  Generate λ offspring via level-switch mutation
  Filter via multi-stage selection (increasing tokens per stage)
  $\mathbf{b}^* \leftarrow$ best offspring by fitness (include parent in final stage)
  if $\mathbf{b}^*$ satisfies $D_{\mathrm{KL}} < \tau$ and $\bar{b}(\mathbf{b}^*) < \bar{b}(\mathbf{b}^{(t)})$ **then**
    $\mathbf{b}^{(t+1)} \leftarrow \mathbf{b}^*$, `stall` $\leftarrow 0$
  else
    `stall` $\leftarrow$ `stall` $+1$
  end if
  if `stall` $\geq$ threshold **then**
    $\mathbf{b}^{(t+1)} \leftarrow \textsc{Curation}(\mathbf{b}^{(t)})$ {Bitwidth-neutral reallocation}
  end if
end while
Return: $\mathbf{b}^{(t)}$

Fitness Function. For an offspring $\mathbf{b}'$ with parent $\mathbf{b}$, let $\Delta b = \bar{b}(\mathbf{b}) - \bar{b}(\mathbf{b}')$ denote the bitwidth reduction. When $\Delta b > 0$, fitness is:

$$f(\mathbf{b}') = \frac{D_{\mathrm{KL}}(\mathbf{b}')}{\Delta b} \cdot \begin{cases} 1 & \text{if } D_{\mathrm{KL}}(\mathbf{b}') \leq \tau \\ \gamma \cdot (D_{\mathrm{KL}}(\mathbf{b}') - \tau) & \text{otherwise} \end{cases} \tag{9}$$

where $\gamma \gg 1$ is a penalty coefficient. Lower fitness is better. Offspring that increase bitwidth ($\Delta b < 0$) receive heavily penalized fitness values.

Adaptive Curation. The inverted formulation introduces a challenge: because the search only accepts bitwidth-reducing mutations, suboptimal early allocations cannot be corrected. When progress stalls, curation performs bitwidth-neutral swaps between equal-sized groups, keeping the average constant while reallocating capacity from insensitive to sensitive groups. This allows the search to escape local optima and continue compressing.

Limitations. The evolutionary search requires orders of magnitude more forward passes than the ILP-based solver and does not scale to large models. We include it here as a reference formulation; all experimental results in the main paper use the sensitivity-based ILP solver.

B Derivation of the Gamma-Squared Variance Law

This appendix provides a self-contained derivation of Lemma 3.2. We proceed in three steps: analyzing quantizer step sizes, deriving quantization noise variance via Bennett's high-rate approximation, and combining the two results.

B.1 Definitions and Preliminaries

Let W be a scalar random variable representing a neural network weight, with probability density function supported on $[L, U]$ where $L < U$. We recall the geometric quantities from Definition 3.1: the dynamic range $R = U - L$, the maximum absolute magnitude $M = \max(|L|, |U|)$, and the centering inefficiency $\gamma = 2M/R$. For distributions that are not symmetric about zero, the symmetric interval $[-M, M]$ strictly contains $[L, U]$, so $\gamma > 1$.

B.2 Step 1: Quantization Step Size Analysis

Consider a uniform scalar quantizer with n discrete levels (typically $n = 2^b$ for bitwidth b). The step size Δ is the distance between adjacent quantization levels.

Asymmetric quantizer. The grid spans exactly $[L, U]$, dividing the range R into $n-1$ intervals:

$$\Delta_{\text{asym}} = \frac{R}{n-1}. \tag{10}$$

Symmetric quantizer. The grid is constrained to be symmetric about zero, spanning $[-M, M]$:

$$\Delta_{\text{sym}} = \frac{2M}{n-1}. \tag{11}$$

Lemma B.1 (Step size scaling)**.** *The symmetric step size relates to the asymmetric step size by* $\Delta_{\text{sym}} = \gamma\, \Delta_{\text{asym}}$.

Proof. Direct computation:

$$\frac{\Delta_{\text{sym}}}{\Delta_{\text{asym}}} = \frac{2M/(n-1)}{R/(n-1)} = \frac{2M}{R} = \gamma. \tag{12}$$

□

B.3 Step 2: Quantization Noise Variance

We analyze the noise characteristics using classical results from quantization theory.

Bennett's approximation vs. Bussgang decomposition. Two classical results characterize quantization noise. Bennett's approximation (Bennett, 1948) shows that, under the high-rate assumption (many quantization levels), the quantization error $e = Q(W) - W$ is approximately uniformly distributed within each bin, $e \sim \mathcal{U}[-\Delta/2, \Delta/2]$. The only requirement is that the probability density of W is approximately constant within each bin, which is satisfied when the number of quantization levels is large relative to the variation in the density. The Bussgang decomposition (Bussgang, 1952) provides a complementary signal-processing perspective, decomposing the quantizer output as $Q(W) = \alpha W + d$ where α is a gain term and d is noise uncorrelated with the input ($\mathbb{E}[Wd] = 0$). This uncorrelation guarantee, however, requires the input W to be Gaussian. Since LLM weight distributions are generally non-Gaussian, we rely on Bennett's result, which applies under the weaker high-rate condition.

Lemma B.2 (Variance of quantization noise)**.** *Under the high-rate assumption, the variance of the quantization noise is* $\sigma^2 = \Delta^2/12$.

Proof. The mean of $e \sim \mathcal{U}[-\Delta/2, \Delta/2]$ is zero. The variance is:

$$\sigma^2 = \int_{-\Delta/2}^{\Delta/2} x^2 \cdot \frac{1}{\Delta}\, dx \tag{13}$$

$$= \frac{1}{\Delta}\left[\frac{x^3}{3}\right]_{-\Delta/2}^{\Delta/2} \tag{14}$$

$$= \frac{1}{\Delta} \cdot \frac{2(\Delta/2)^3}{3} \tag{15}$$

$$= \frac{\Delta^2}{12}. \tag{16}$$

□

B.4 Step 3: Proof of the Gamma-Squared Law

We combine Lemma B.1 and Lemma B.2 to establish the main result.

Proof of Lemma 3.2. Applying Lemma B.2 to each quantizer:

$$\sigma^2_{\text{asym}} = \frac{\Delta^2_{\text{asym}}}{12}, \qquad \sigma^2_{\text{sym}} = \frac{\Delta^2_{\text{sym}}}{12}. \tag{17}$$

Substituting $\Delta_{\text{sym}} = \gamma\, \Delta_{\text{asym}}$ from Lemma B.1:

$$\sigma^2_{\text{sym}} = \frac{(\gamma\, \Delta_{\text{asym}})^2}{12} \tag{18}$$

$$= \gamma^2 \cdot \frac{\Delta^2_{\text{asym}}}{12} \tag{19}$$

$$= \gamma^2\, \sigma^2_{\text{asym}}. \tag{20}$$

□

The geometric penalty γ imposed by symmetric quantization on asymmetric weight distributions therefore leads to a quadratic increase in quantization noise variance. For a single linear layer $\mathbf{y} = W\mathbf{x}$, the output perturbation due to quantization is $\boldsymbol{\delta} = E\mathbf{x}$ where $E = Q(W) - W$. The expected squared error per output unit is $\sigma^2\|\mathbf{x}\|^2$. Since symmetric and asymmetric quantization are applied to the same weights with identical inputs, $\|\mathbf{x}\|^2$ cancels in the ratio, giving $\mathbb{E}[\|\boldsymbol{\delta}_{\text{sym}}\|^2]/\mathbb{E}[\|\boldsymbol{\delta}_{\text{asym}}\|^2] = \gamma^2$. This provides the analytical foundation for the empirical observations in Section 3.1, where symmetric quantization exhibits consistently higher flip rates and lower EAR.

C Additional Experiments and Results

We present additional experiments that support and extend the main results. Appendix C.1 shows the per-layer sensitivity landscape and how it translates into bitwidth assignments. Appendix C.2 quantifies the asymmetric-vs.-symmetric gap on a task-level benchmark. Appendix C.4 measures per-layer kernel speedups across bitwidths. Appendix C.5 characterizes how distributional shift (KL) relates to zero-shot accuracy, connecting the distribution-lossless (DL) and task-lossless (TL) operating points. Appendix F reports exact effective bits per parameter accounting for all overheads. Appendix F.1 inspects token-level divergence under DL compression. Appendix G.1 gives the full sweep of quantization configurations, formats, and allocation strategies.

C.1 Sensitivity Estimates and Bitwidth Assignments

Figures 5 and 6 illustrate the two stages of the SLQ pipeline for Qwen3-8B: sensitivity estimation and bitwidth allocation. Both methods agree on the broad sensitivity structure (attention K/V most sensitive, MLP least), but differ in how they capture it, which affects the final allocation.

Figure 5 shows the continuous per-layer EAR sensitivity coefficients from linear estimation (Section 3.3). The coefficients reflect each layer's independent contribution to output degradation: K and V projections are consistently the most sensitive, while MLP gate/up projections and the output projection are less so. Early and late layers tend to be more sensitive than middle layers.

Figure 6 shows the discrete bitwidth assignment produced by the ILP solver using Shapley-based sensitivity estimates, targeting distribution-lossless compression (5.70 bits average, EAR = 99.0%). Because Shapley values account for cross-layer interactions during permutation sampling, the resulting allocation can differ from what linear estimates alone would produce. The solver assigns 8 bits to the most sensitive K/V projections, 6–7 bits to Q and output projections, and 4–5 bits to the more robust MLP layers.

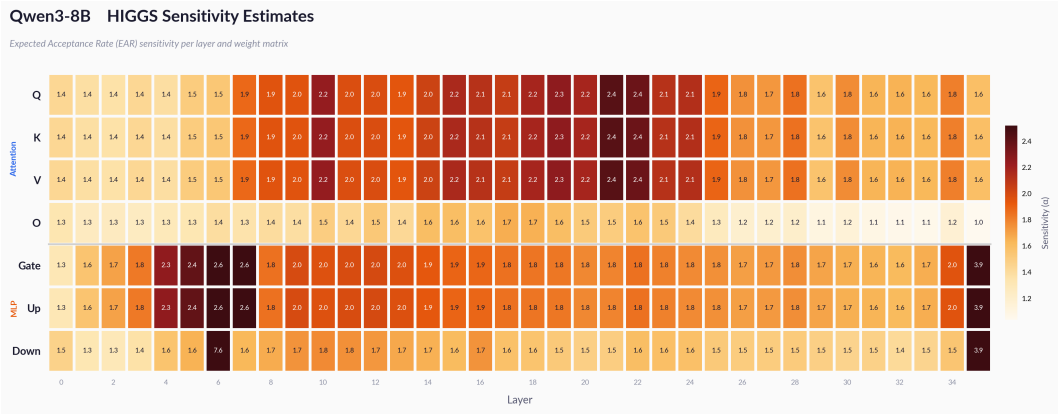


Figure 5: Per-layer EAR sensitivity coefficients from linear estimation for Qwen3-8B. Continuous values represent each layer's independent contribution to output degradation.

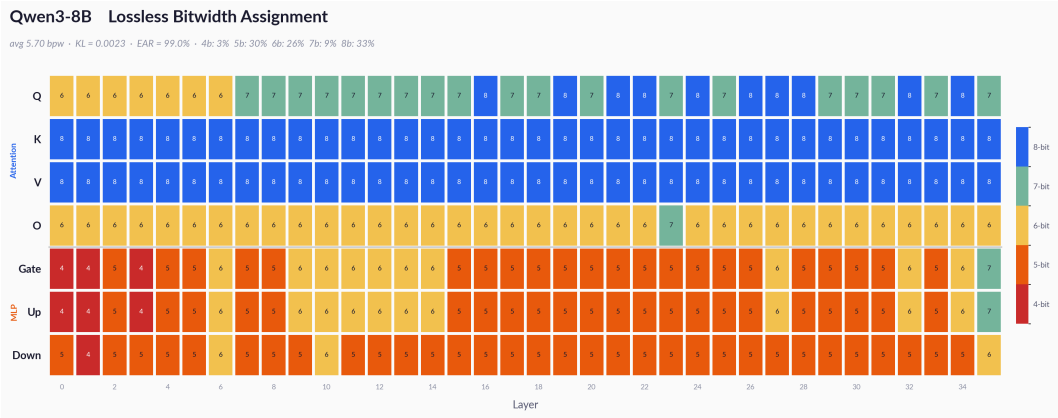


Figure 6: Bitwidth assignment from Shapley-based allocation for Qwen3-8B DL configuration (5.70 avg bits, EAR = 99.0%, KL = 0.0023). The solver assigns higher bitwidths to sensitive attention layers and lower bitwidths to robust MLP layers.

C.2 Asymmetric vs. Symmetric: Additional Results

Table 4 provides the full MMLU-Pro breakdown referenced in Section 3.1. The task reduces to a single-token decision (A/B/C/D), so every decision flip directly changes the answer. The model is instructed to output only the final answer token; overall scores are therefore lower than with chain-of-thought reasoning.

Table 4: MMLU-Pro zero-shot (12,032 samples), uniform 4-bit, Llama-3.1-8B-Instruct.

Metric	FP16	Asymmetric	Symmetric
Accuracy	18.09%	**15.99%**	14.92%
Recovery rate	100%	**88.4%**	82.5%
Mean margin on flip	0	**0.162**	0.175

The asymmetric–symmetric gap is most pronounced for distribution-lossless (DL) compression, where EAR is the target metric. For task-lossless (TL) compression, symmetric quantization may still suffice since benchmark scores absorb additional token flips without

measurably affecting aggregate accuracy (Section 3.1). Full EAR trajectories across bitwidths and additional results on margin at disagreement appear in Appendix G.1.

C.3 Comparison to Stretched Elastic Quantization

Stretched Elastic Quantization (SEQ) (Liu et al., 2025b) addresses a *level-allocation* inefficiency: it spaces levels evenly across the span, which omits a zero level at even level counts (2-bit: $\{\pm 0.5, \pm 1.5\}$). The γ^2 law concerns a different, *centering* inefficiency, and the two are independent. SEQ learns only the scale α, which rescales the grid but never shifts it off zero, so a SEQ grid remains centered and cannot fit itself to an off-center $[L, U]$; only a zero-point realizes the finer step of Lemma 3.2. SEQ therefore gives no relief on the centering axis, and the choice between the two is not a trade-off. Consistently, ParetoQ applies SEQ only at ternary and 2-bit precision, where scarce levels make their allocation the binding constraint, and reverts to the zero-inclusive LSQ grid at 3 and 4 bit.

C.4 Speedup with Humming Kernels

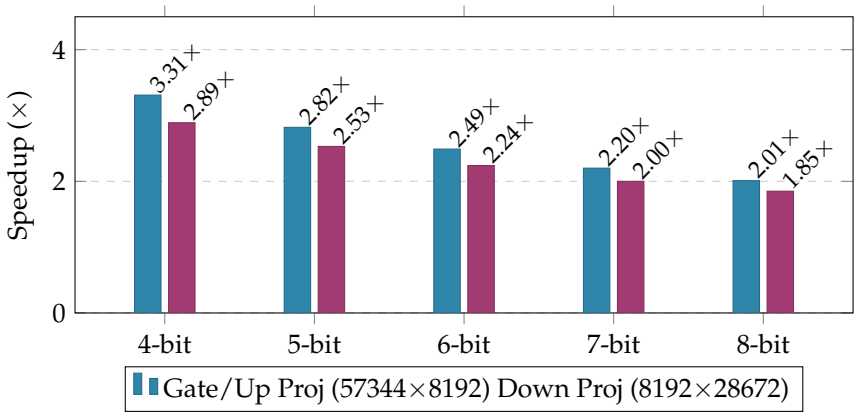


Figure 7: Speedup relative to FP16 baseline for different quantization bitwidths, when executing layers of Llama-3-70B.

C.5 KL Divergence and Zero-Shot Performance

To establish the relationship between output fidelity and downstream task performance, we conduct controlled experiments that systematically vary the degree of distributional shift from the original model. We inject uniform noise into model weights at increasing levels, where noise magnitude is scaled by each weight matrix's standard deviation to approximate quantization error characteristics. For each noise level, we compute KL divergence between the perturbed and original model's output distributions, then evaluate zero-shot accuracy across five task categories: Reasoning (ARC, HellaSwag, WinoGrande, PIQA), Knowledge (MMLU, TriviaQA), Math (GSM8K, MATH), Code (HumanEval, MBPP), and Retrieval (NeedleBench).

Figure 9 presents degradation curves across six models spanning different scales and architectures. The results reveal a consistent pattern: accuracy remains stable while KL divergence remains small, then degrades as divergence increases. The specific onset of degradation varies with model scale and task category, with Retrieval tasks demonstrating the greatest robustness and Math and Code proving more sensitive to distributional shift. Crucially, all models maintain baseline accuracy when KL divergence stays below 0.01. This establishes the link between distribution-lossless (DL) and task-lossless (TL) compression: a model that is DL (high distributional fidelity, as measured by EAR or KL) is also TL (preserves zero-shot task accuracy). The converse does not hold: a model can be TL at a lower bitwidth without meeting the stricter DL criterion, since benchmark scores are less sensitive than distributional metrics (Section 3.3).

Figure 8 zooms into the near-lossless regime on a linear scale, complementing the log-scale overview of Figure 1b. The approximately linear decline of accuracy with KL divergence in this range is the property exploited by single-point calibration (Section 3.3): it is what allows a single anchor measurement to determine the slope α, and hence the KL threshold corresponding to any target recovery.

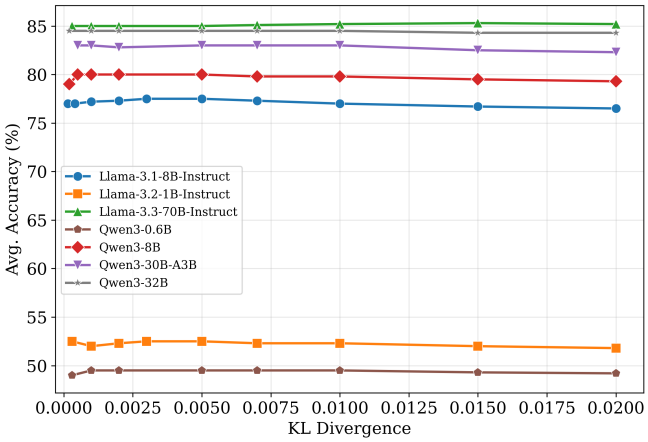


Figure 8: Near-lossless regime on a linear scale. Accuracy declines approximately linearly with KL divergence, which single-point calibration exploits to derive a KL threshold from one benchmark measurement.

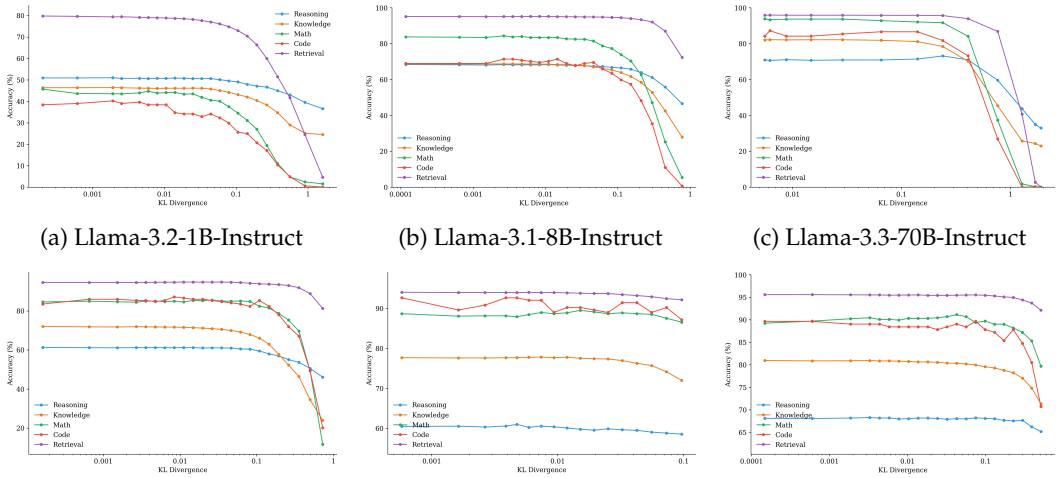
Table 5: KL divergence on Llama-3.1-8B-Instruct. Methods above the rule operate at uniform 4-bit precision; SLQ uses non-uniform scalar integer allocation across $\{4, 5, 6, 7, 8\}$ bits (Section 3). GPTQ, QTIP and SLQ are measured by us on the released checkpoints under an identical evaluation setup. †: value as reported by Tseng et al. (2025), which we did not reproduce on this model; our own YAQA measurements on Llama-3.3-70B are in Table 2.

Method	Bits	KL ↓	EAR ↑
GPTQ (asym, g128)	4.16	0.028	0.938
QTIP	∼4	0.017	0.952
YAQA†	∼4	0.028	–
SLQ	5.59	**0.004**	**0.994**

D Comparison to State-of-the-Art Quantizers

This appendix details the evaluation behind Table 2. We compare against two strong per-layer quantizers that target the near-lossless frontier: QTIP (Tseng et al., 2024b), a trellis-coded quantizer, and YAQA (Tseng et al., 2025), which uses full-model-KL-aware Hessian sketching. Both are orthogonal to SLQ in the sense that they replace the per-layer quantizer while leaving the allocation question open; our framework would apply on top of either, at the cost of substantially heavier system support.

Setup. We evaluate the publicly released checkpoints for Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct rather than quoting published numbers, so that tasks, prompts, sampling parameters, seeds and the KL/EAR calibration set match those used for our own models exactly (Section 4). Task mean recovery is the average over the seven benchmarks of Table 1, relative to the BF16 baseline, under non-greedy decoding averaged over 3 seeds. KL and EAR are computed on the same calibration positions as for SLQ, restricted to the top-K=10 tokens.



(a) Llama-3.2-1B-Instruct (b) Llama-3.1-8B-Instruct (c) Llama-3.3-70B-Instruct

(d) Qwen3-8B (e) Qwen3-30B-A3B (f) Qwen3-32B

Figure 9: Zero-shot accuracy versus KL divergence across task categories for different model architectures and sizes. Accuracy remains stable until KL divergence exceeds approximately 0.01: distribution-lossless (DL) compression implies task-lossless (TL) compression, but TL can be achieved at lower bitwidths since benchmarks are less sensitive than distributional metrics. Larger models tolerate greater distributional shift before degradation. The x-axis uses logarithmic scale.

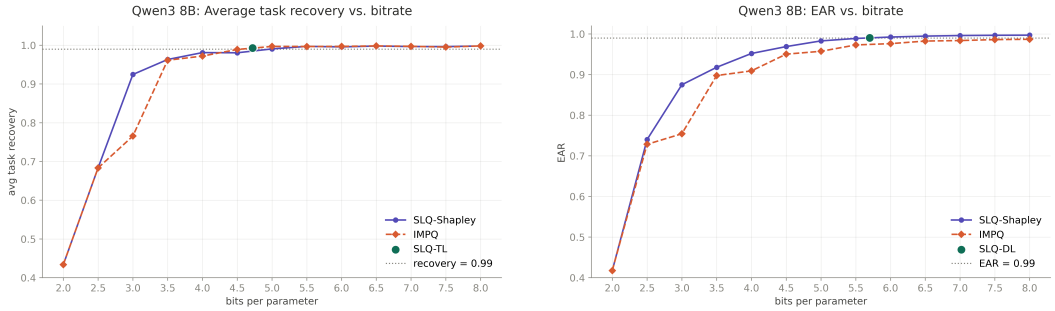
Findings. Neither method reaches the distribution-lossless regime at any tested uniform bitwidth. On Llama-3.1-8B, QTIP at 4-bit attains 97.94% mean recovery, which misses the task-lossless threshold of 99% outright, with EAR 0.9519; at 3-bit the model is visibly lossy (96.00%, EAR 0.9165). On Llama-3.3-70B, both QTIP (99.9%) and YAQA (99.5%) clear the task-lossless bar at 4-bit, but their EAR values of 0.9544 and 0.9622 remain far below the DL threshold of 0.99. SLQ reaches task-losslessness on the same model at 3.59 bpp, i.e. it clears the same threshold at a lower bitwidth, and reaches DL at 6.55 bpp.

These quantizers are not weak: at a fixed 4-bit budget they are considerably stronger than GPTQ, which is why we adopt them as the reference point. The uniform bit budget is instead selected without reference to any fidelity target, and therefore overshoots it on one model while undershooting it on another. Choosing the bitwidth from the fidelity target instead, which is what SLQ does, is what makes the near-lossless operating point reachable at all.

D.1 Comparison to Non-Uniform Allocators

Figure 10 accompanies the discussion in Section 4.1. We compare SLQ against IMPQ (Zhao et al., 2025) on Qwen3-8B, sweeping both methods across bitrates and recording average task recovery and EAR at each operating point. IMPQ is run in its published configuration, which restricts allocation to the binary choice $\{4, 8\}$.

The two panels separate the two effects. On average task recovery the methods are close above $\sim$4 bpp, both saturating near the BF16 baseline; benchmark accuracy simply cannot resolve configurations at this level of fidelity, which is the motivation for adopting EAR as the primary search objective. On EAR the curves separate, and IMPQ's binary allocation does not reach the DL threshold anywhere in the tested range, consistent with the bitwidth-range ablation in Section 4.1. Beyond the gap itself, neither IMPQ nor the other non-uniform allocators provide a mechanism to identify where losslessness begins: locating the green operating points with a budget-driven method requires sweeping the entire curve and running benchmark evaluations at every budget, whereas SLQ obtains them directly from a fidelity constraint evaluated on calibration data.



(a) Average task recovery vs. bitrate.

(b) EAR vs. bitrate.

Figure 10: SLQ versus the IMPQ non-uniform allocator on Qwen3-8B. Green dots mark the SLQ-TL (a) and SLQ-DL (b) operating points selected by our fidelity-targeted search; dotted lines mark the corresponding thresholds. Task recovery saturates for both methods, while EAR separates them.

E Scope and Limitations

This appendix expands on the three limitations summarized in the conclusion.

Calibration cost. Multi-bitwidth Shapley estimation runs $O(P \cdot M \cdot |\mathcal{B}|)$ forward passes, against $O(M \cdot |\mathcal{B}|)$ for the linear estimator of Malinovskii et al. (2025). The ratio of 10–100$\times$ is independent of model size, but it multiplies a per-pass cost that is not. On an 8B model this amounts to roughly 2.5 h per bitwidth on a single A6000, or $\sim$15 h across the six non-reference bitwidths; the games are independent, so the wall-clock cost divides by the number of available GPUs. At 100B+ scale the absolute cost becomes prohibitive and linear estimation is the appropriate choice.

The trade-off is compute against bitwidth rather than compute against fidelity. Both estimators drive the search to the same fidelity target, so both produce statistically-lossless models; Shapley's advantage is that the resulting model is slightly smaller, an advantage concentrated in the 5.5–6.5 bpp band where allocation decisions are tightest (Appendix G.1). Linear estimation therefore costs bitwidth efficiency, but not losslessness.

Weight-and-activation quantization. SLQ is agnostic to activation quantization. Sensitivity estimation already incorporates activation noise, and neither the ILP formulation nor the EAR objective changes when activations are quantized, so W+A requires no algorithmic modification. We nonetheless report W+A only at small scale, because hardware-supported activation formats remain far more limited than weight formats: the available options force a coarser grid than the full $\{2, \ldots, 8\}$ range on which near-lossless compression depends (Appendix G.1). This is a property of the current kernel ecosystem rather than of the method, and the restriction lifts as activation format support matures.

Choice of per-layer quantizer. We use GPTQ throughout because it is well-tuned and the de facto standard for vLLM deployment. The γ^2 law and the TL/DL framework apply to any layer-wise quantizer, including vector- and trellis-coded methods. In the near-lossless regime, non-uniform allocation and grid asymmetry are the dominant factors; a higher-fidelity per-layer quantizer helps at the margin, as Table 2 indicates, but does not by itself move a uniform configuration across the losslessness boundary.

Attention architectures. All evaluated models (the Qwen3 family and Llama-3.3-70B) use grouped-query attention. Attention projections account for a small fraction of total parameters, so differences in per-group sensitivity structure between GQA and multi-head attention do not materially affect the resulting average bitwidth or throughput.

F Mixture-of-Experts Models

To test whether the SLQ pipeline transfers to sparsely-activated architectures, we evaluate Qwen3.5-35B-A3B (35B total parameters, 3B active), the successor to the MoE model appearing in Figure 9. We apply the pipeline without modification: expert and shared layers are partitioned into groups exactly as dense layers are, sensitivity is estimated with the same multi-bitwidth Shapley procedure, and allocation uses the same ILP under the same fidelity constraints. No MoE-specific handling (expert-aware grouping, routing-aware calibration, or per-expert bit budgets) was introduced.

Table 6: SLQ on Qwen3.5-35B-A3B (35B total, 3B active). Evaluated with the LM Evaluation Harness under its log-likelihood protocol, so absolute scores are not directly comparable to the generative results of Table 1; recovery rates are. Both configurations lie within sampling variance of the BF16 baseline on every benchmark.

Task	BF16	SLQ-TL (3.73 bpp)	SLQ-DL (5.75 bpp)
ARC-Easy	0.8392	0.8325	0.8405
ARC-Challenge	0.5887	0.5836	0.5904
HellaSwag	0.6234	0.6225	0.6241
PIQA	0.8281	0.8264	0.8254
WinoGrande	0.7388	0.7474	0.7451
MMLU	0.8266	0.8265	0.8276
GSM8K	0.8491	0.8901	0.8355
Mean recovery	–	**99.8%**	**100.1%**

Table 6 reports per-task results over the seven benchmarks we ran for this model. SLQ-TL reaches 99.8% mean recovery at 3.73 bpp and SLQ-DL reaches 100.1% at 5.75 bpp. The per-task spread behaves as it does for the dense models: individual benchmarks fluctuate in both directions around the baseline (GSM8K is the widest, in line with its known run-to-run variance under stochastic sampling), while the aggregate stays within sampling variance. The resulting bitwidths are also consistent with the dense results of Table 1, falling inside the 3.3–4.7 bpp TL and 5.0–6.6 bpp DL ranges observed there, which suggests the losslessness thresholds are governed by the fidelity target rather than by whether computation is dense or sparsely routed.

Actual average bits per parameter. We report the exact average number of bits per parameter to enable precise comparisons at a given effective bitwidth. For each configuration, the reported value accounts for the true storage cost of every layer, including all quantization-related overheads. As a result, the averages reflect the realized per-parameter bitwidth rather than the nominal setting.

Table 7: Actual bitwidth by format and nominal bitwidth

	INT (groupsize 128)		NVFP/FP (groupsize 16)	
Nominal	Asym	Sym	Asym	Sym
4	4.156	4.125	5.0	4.5
5	5.156	5.125	–	–
6	6.156	6.125	–	–
7	7.156	7.125	–	–
8	8.156	8.125	8.0	8.0

F.1 Token-Level Divergence Analysis

To visualize the practical impact of distribution-lossless (DL) compression, we perform a token-level comparison between the base Qwen3-8B (FP16) and the SLQ-DL quantized

model (5.70 bpw, EAR = 99%). We generate responses with the base model using greedy decoding, then teacher-force the identical token sequence through the EAR model and record every position where the EAR model's greedy next-token prediction differs. Divergent tokens appear highlighted as `base` with the EAR model's alternative shown as a [blue] superscript.

Methodology. The goal of this analysis is to measure how often the quantized model would choose a different token than the base model at each generation step, i.e. how frequently quantization causes the model to "start diverging" from the original output. To measure this cleanly, we use *teacher-forced comparison*: we first generate a full response from the base model using greedy decoding, then feed that exact token sequence as input to the EAR model and record, at every position, whether the EAR model's greedy top-1 prediction matches the base model's token. Crucially, the EAR model is always conditioned on the base model's actual tokens, not its own previous predictions. This is necessary because in free (autoregressive) generation, a single divergent token shifts the entire continuation, making it impossible to attribute later differences to quantization error versus the natural consequence of conditioning on a different prefix. Teacher-forcing isolates the per-position effect of quantization: each reported divergence reflects a position where quantization alone caused the model to prefer a different token, independent of any earlier divergences.

General statistics. We evaluate 15 diverse prompts spanning code generation, mathematical derivation, essay writing, logic puzzles, and everyday questions, producing 6,531 tokens in total. Of these, 6,407 tokens (98.1%) are identical between both models; only 124 tokens (1.9%) diverge. Per-prompt agreement ranges from 96.1% to 99.6% with a median of 98.2%. Divergences are distributed roughly uniformly across token positions (mean position 235 out of 512), confirming that quantization errors do not compound over longer sequences. All observed divergences are minor stylistic variations—synonym swaps, formatting changes, or punctuation differences—and none alter factual content.

Most frequently divergent tokens. Since 124 divergent tokens are distributed across 6,531 generated tokens, no single token dominates: the most frequent base tokens appear at most 3 times. Table 8 lists the 10 most common base tokens involved in divergences together with their most frequent substitutions.

Table 8: Most frequently divergent base tokens across 15 prompts. Each row shows a base-model token that the EAR model replaces, its occurrence count, and representative substitution(s).

Base token	Count	EAR substitution(s)
`The`	3	`###, **, Inside`
`(`	3	`of, ,`
`the`	3	`a, which, it`
`and`	2	`through, ,`
`genetic`	2	`instructions, the`
`of`	2	`on, :`
`.\n`	2	`of, into`
`,`	2	`to, \n`
`to`	2	`., --`
`**`	2	`logic, the`

Per-prompt summary. Table 9 lists agreement for all 15 prompts.

Representative examples. Three examples follow, chosen to cover different domains. All three exceed 96% token-level agreement.

Example 1 *(normal text, 364 tokens, 13 divergent, 96.4% agreement)*

Table 9: Token-level agreement between base Qwen3-8B (FP16) and the 99% EAR model (5.70 bpw) across 15 diverse prompts with greedy decoding.

Prompt	Type	Tok.	Agr.	Div.
Recipe for cookies	Normal	268	99.6%	1
Photosynthesis	Q&A	138	99.3%	1
Binary search tree	Code	512	99.2%	4
Determinant intuition	Reason.	512	99.0%	5
SQL top-5 customers	Code	446	98.9%	5
Geometric series sum	Reason.	512	98.8%	6
Merge intervals	Code	448	98.4%	7
Wolf, goat, cabbage	Logic	512	98.2%	9
FDR vs. Reagan	Essay	512	98.0%	10
Python vs. Java	Code	512	97.8%	11
Themes of isolation	Essay	512	97.7%	12
Byzantine Generals	Reason.	512	97.7%	12
Romeo and Juliet	Summ.	259	96.9%	8
Car engine for kids	Q&A	364	96.4%	13
mRNA vaccines	Reason.	512	96.1%	20
Total (15 prompts)		**6,531**	**98.1%**	**124**

Explain how a car engine works to a 10-year-old.

```
Sure!  Let me explain how a car engine works in a way that's easy to understand, like
you're  learning 10 years old.  Imagine your car is like a big toy that runs on gas, like
the kind you put in your gas tank.  Inside the car, there's something called an **engine**.
The engine is like the heart of the car ---it · helps it move.  The Inside engine has a bunch
of small parts that move around, kind of like the wheels on a toy car.  These parts are
called ** c P ylinders**.  Each cylinder is like a tiny  cylinder chamber (that like's why
they're called cylinders!)  that where moves up and down.  Here's what happens:  1.  **Gas
is put in**:  You put gas in the tank, and it goes to the engine.  2.  **Spark plug lights
it up**:  Inside the cylinder, there's a spark plug that lights the gas on fire, like a
tiny fire cr in acker.  3.  **Fire makes power**:  When the gas burns, it makes a  lot big
of heat and pressure.  This pressure pushes the cylinder down, which turns a rod connected
to the wheels.  4.  **Wheels turn**:  The rod turns a big wheel called the **crankshaft**,
which makes the wheels of the car turn.  5.  **Repeat **: over The process repeats over
and over again, like a big, powerful  heartbeat toy, and that's how the car moves!  So,
the engine is like a big, powerful machine that uses gas to make the car go.  It's kind
of like a magic box that turns gas into motion!  Want to know what happens when the car
stops?  Then The the engine stops, and the car doesn't move anymore.  But when you press
the gas pedal,  the it engine starts working again!  Does that make sense?
```

Example 2 *(mathematical derivation, 320 tokens shown, 2 divergent, 98.8% agreement)*

Derive the formula for the sum of an infinite geometric series. Under what conditions does the series converge?

```
To **derive the formula for the sum of an infinite geometric series**, we start with
the definition of a geometric series.  --- ### **Definition of a Geometric Series** A
**geometric series** is a series of the form:  $$ S = a + ar + ar^2 + ar^3 + \cdots $$
Where:  - $ a $ is the **first term**, - $ r $ is the **common ratio** between successive
terms.  This series continues **indefinitely**, so it is an **infinite geometric series**.
--- ### **Der  $  as the Formula** Lets denote the sum of the infinite geometric series as
$ S $:  $$ S = a + ar + ar^2 + ar^3 + \cdots $$ Now, multiply both sides of the equation
by the common ratio $ r $:  $$ rS = ar + ar^2 + ar^3 + ar^4 + \cdots $$ Now subtract the
second equation from the first:  $$ S - rS = (a + ar + ar^2 + ar^3 + \cdots) - (ar + ar^2 +
ar^3 + \cdots) $$ On the right-hand side, all terms cancel except the first term $ a $:  $$
S(1 - r) = a $$ Now solve for $ S $:  $$ S = \frac{a}{1 - r} $$ --- ### **Final valid only:**
$$ \boxed{S = [...]
```

Example 3 *(distributed systems/reasoning, 280 tokens shown, 7 divergent, 97.7% agreement)*

Explain the Byzantine Generals Problem in distributed computing. Why is it impossible to solve with one traitorous general?

```
The **Byzantine Generals Problem** is a classic problem in **distributed computing** that
illustrates the challenges consensus ** achieving **consensus** in a network of processes
(or "generals") when is was of them may story historical **faulty** or **malicious**.  It
was introduced by **Leslie Lamport, Robert Shostak, and Marshall Pease** in a 1982 paper
titled *"The Byzantine Generals Problem."* --- ## **The Problem:  The Byzantine Generals**
Imagine a group of generals, each commanding a division of the army, surrounding a city.
They must decide whether to **attack** or **retreat**.  However, they can only communicate
with each other through **messengers**, and some of the generals may be **traitors** (i.e.,
they may send false messages or act against the interests of the group).  The goal is for
all loyal generals ** by **agree on the same plan** (attack or retreat), even if some are
traitors.  The problem is to determine whether this is possible, communication messages
under what conditions.  --- ## **Key Assumptions** - There are **n** generals (processes).
- At most **m** of them are **traitors** (faulty or malicious).  - There Nodes are is
**synchronous** (messages are delivered in a finite [...]
```

G Thinking Token Inflation Under Quantization

We measure whether quantized models generate more thinking tokens (`<think>...</think>`) than the BF16 baseline to reach the same answer on Qwen3-8B, GSM8K (1319 problems).

Setup. Inference uses vLLM (v0.17.1) with identical sampling across all configs (temperature=0.6, top_p=0.95, top_k=20). Each query is run 3 times (seeds 43, 44, 45) for within-query variance, yielding 3957 generations per config. Maximum generation length is 8192 tokens with context length 16384. Quantized weights are fake-quantized (dequantized) with asymmetric integer quantization, group size 128. Hardware: NVIDIA L40S (46 GB).

Aggregate statistics. Table 10 reports per-config statistics over all 3957 generations.

Table 10: Thinking token statistics on GSM8K (Qwen3-8B, 1319 problems $\times$ 3 seeds).

Config	Bits/param	Mean	Median	Std	Accuracy	Truncated
BF16	16.00	2051	1548	1509	94.1%	40
SLQ-DL	5.70	2051	1555	1518	93.8%	53
SLQ-TL	4.72	2097	1619	1504	94.7%	50
Uniform-4	4.00	2111	1627	1508	94.6%	54

Paired comparison vs BF16. Per-query means are computed by averaging over 3 seeds, and compared to BF16 via paired Wilcoxon signed-rank test (Table 11).

Table 11: Paired thinking token comparison vs BF16 (N=1319 queries).

Config	Mean diff	Median ratio	Wilcoxon p	Cohen's d
SLQ-DL	-1	0.995	8.6×10^{-1}	-0.001
SLQ-TL	$+45$	1.037	3.7×10^{-7}	$+0.076$
Uniform-4	$+59$	1.036	5.2×10^{-6}	$+0.097$

Jointly-correct queries. We restrict to queries where both BF16 and the quantized model answer correctly on all 3 runs, testing whether inflation persists even when both models reach the right answer (Table 12).

Table 12: Thinking tokens on jointly-correct queries.

Config	N queries	BF16 mean	Quant mean	Diff	Wilcoxon p
SLQ-DL	1195	1800	1798	-2	6.3×10^{-1}
SLQ-TL	1195	1789	1839	$+50$	2.4×10^{-7}
Uniform-4	1192	1799	1861	$+62$	2.0×10^{-6}

Thinking tokens by correctness. Wrong answers consume 2.4–2.6× more thinking tokens across all configs (Table 13).

Table 13: Mean thinking tokens by answer correctness.

Config	Correct mean	Wrong mean	Ratio (wrong/correct)
BF16	1893	4564	2.41
SLQ-DL	1882	4605	2.45
SLQ-TL	1934	5003	2.59
Uniform-4	1945	4987	2.56

Within-query variance. Coefficient of variation (CV = std / mean) across 3 runs of the same query (Table 14). The 2–3% inflation from lossy quantization is small relative to natural run-to-run variance (∼19%), but statistically significant with N=1319 paired comparisons.

Table 14: Within-query coefficient of variation across 3 seeds.

Config	Mean CV	Median CV
BF16	18.8%	16.9%
SLQ-DL	18.8%	16.9%
SLQ-TL	19.3%	17.3%
Uniform-4	19.5%	17.3%

Token budget breakdown. The inflation is entirely in the thinking portion; non-thinking output length is stable across configs (Table 15).

Table 15: Mean token budget breakdown per generation.

Config	Think	Non-think	Total	Think %
BF16	2051	305	2356	87.1%
SLQ-DL	2051	306	2356	87.0%
SLQ-TL	2097	302	2399	87.4%
Uniform-4	2111	286	2397	88.1%

Thinking token percentiles. Table 16 shows the distribution across generations.

Table 16: Thinking token percentiles across all generations.

Config	P10	P25	P50	P75	P90	P95	P99
BF16	803	1085	1548	2462	3993	5362	8179
SLQ-DL	802	1073	1555	2466	3944	5371	8192
SLQ-TL	841	1101	1619	2490	4061	5337	8192
Uniform-4	831	1117	1627	2554	4062	5408	8192

G.1 Detailed Quantization Results

This section presents comprehensive quantization results across bitwidths, number formats, and quantization schemes. All results are evaluated on Llama-3.1-8B-Instruct using 512 calibration samples. The metrics reported here (EAR, KL, Margin, PPL) are distributional fidelity measures used for distribution-lossless (DL) compression. Task-lossless (TL) compression targets benchmark recovery instead and can be achieved at lower bitwidths; see the main results in Section 4.1.

Metrics. We report four key metrics for quantization quality:

- **EAR**: The expected acceptance rate, measuring the overlap in output probability mass between BF16 and quantized models. This equals $\sum_k \min(p_k, q_k)$ where p and q are the top-10 token distributions. Higher EAR indicates closer alignment to the original output distribution.
- **Margin at Disagreement**: The average probability gap on tokens where the quantized and BF16 models disagree on their top prediction. Lower margin means flips occur only on close calls, indicating smaller quantization error.
- **KL Divergence**: The KL divergence between BF16 and quantized output distributions on the calibration set.
- **Perplexity (PPL)**: The perplexity of the quantized model relative to BF16, computed from the calibration set.

Format and Bitwidth Notations. All tables use asymmetric integer quantization (Asym) or symmetric integer quantization (Sym) with group size 128. Bitwidth values (e.g., "4.50") include the overhead from scales and zero-points. The format notation "4-8" means only bitwidths 4 and 8 are available for allocation, while "4-5-6-7-8" means all intermediate bitwidths are available.

Uniform INT4 baselines. Table 17 shows uniform 4-bit integer quantization results for asymmetric and symmetric schemes. These serve as single-bitwidth reference points.

Weights+Activations (W+A) Quantization. Table 18 compares weight-only (W) quantization with combined weights+activations (W+A) quantization, both obtained via evolutionary search with adaptive curation (Appendix A.4) for Llama 3.1 8B Instruct. Weight-only quantization at 5.42 bits achieves excellent recovery across all tasks (97.6%–100.5%), with GSM8K, PIQA, and Winogrande exceeding baseline accuracy. Quantizing both weights and activations at a comparable constraint requires a higher bitwidth of 6.50–6.97 bits. At 6.97 bits (Max KL 0.01), the W+A configuration recovers 99.4%–100.1% on all tasks. This shows that activations are more sensitive to quantization than weights, requiring additional precision to maintain the same quality level, but near-lossless W+A quantization remains feasible.

Table 17: INT4 Reference Quantization Results

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform (Asym)	4.156	0.958	0.113	0.0369	0.956
Uniform (Sym)	4.156	0.946	0.148	0.0576	0.926

Integer quantization with binary bitwidth range. Tables 19 and 20 show integer asymmetric and symmetric quantization when only bitwidths 4 and 8 are available for allocation. Uniform baselines at each single bitwidth are included for reference. Linear and Shapley rows show non-uniform allocations at matched average bitwidths.

Integer quantization with full bitwidth range. Tables 21 and 22 show integer asymmetric and symmetric quantization with the full bitwidth set $\{4, 5, 6, 7, 8\}$. The additional intermediate bitwidths give the solver finer-grained control over per-group allocation.

Table 18: Weights+Activations Quantization Results (Evolutionary Search with Adaptive Curation) for Llama 3.1 8B Instruct

Method	Type	Max KL	BW	PPL	HellaSwag ↑	Winogrande ↑	PIQA ↑	GSM8K ↑	ARC-C ↑
FP16	–	–	16	9.10	57.41	67.48	79.76	83.17	52.56
Evo Search	W	0.0125	5.42	9.35	57.37 (99.9%)	67.80 (100.5%)	80.20 (100.5%)	83.55 (100.5%)	51.28 (97.6%)
Evo Search	W+A	0.015	6.50	9.37	57.02 (99.3%)	67.25 (99.6%)	79.38 (99.5%)	82.03 (98.6%)	51.02 (97.1%)
Evo Search	W+A	0.010	6.97	9.33	57.16 (99.6%)	67.48 (100.0%)	79.81 (100.1%)	83.02 (99.8%)	52.22 (99.4%)

Distribution-lossless (DL) constraint-based allocation. Table 23 shows integer asymmetric results from the binary-search solver targeting a DL quality constraint (EAR threshold) rather than a fixed average bitwidth. Results are shown for both the binary $\{4, 8\}$ and full $\{4, 5, 6, 7, 8\}$ bitwidth ranges.

Floating-point quantization. Tables 24 and 25 use a custom floating-point format based on NVFP4 with an added zeropoint for the asymmetric variant. Only bitwidths 4 and 8 are available. These tables complement the integer results above by showing the same asymmetric vs. symmetric comparison in the FP domain.

Table 19: INT Asymmetric Quantization Results (4-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	4.16	0.938	0.148	0.0277	0.970
Uniform	5.16	0.964	0.083	0.0068	0.992
Uniform	6.16	0.988	0.029	0.0017	0.998
Uniform	7.16	0.989	0.023	0.0004	1.000
Uniform	8.16	0.992	0.021	0.0001	1.000
4.5-bit					
Linear	4.50	**0.967**	**0.051**	**0.0196**	0.978
Shapley	4.50	0.952	0.130	0.0199	**0.979**
5.5-bit					
Linear	5.50	**0.975**	**0.047**	0.0145	**0.984**
Shapley	5.50	0.975	**0.047**	**0.0143**	0.984
6.0-bit					
Linear	5.99	0.977	0.051	0.0126	0.986
Shapley	5.99	**0.981**	**0.047**	**0.0122**	**0.986**
6.5-bit					
Linear	6.50	0.983	0.043	0.0098	0.989
Shapley	6.50	**0.984**	**0.029**	**0.0091**	**0.990**
7.0-bit					
Linear	6.98	**0.986**	0.043	0.0073	**0.992**
Shapley	6.98	0.984	**0.029**	**0.0069**	0.991

Table 20: INT Symmetric Quantization Results (4-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	4.12	0.938	0.145	0.0344	0.957
Uniform	5.12	0.964	0.083	0.0087	0.991
Uniform	6.12	0.985	0.051	0.0022	0.997
Uniform	7.12	0.989	0.023	0.0006	0.999
Uniform	8.12	0.997	0.005	0.0001	1.000
4.5-bit					
Linear	4.50	**0.955**	**0.124**	**0.0236**	**0.971**
Shapley	4.50	0.953	0.132	0.0250	0.970
5.5-bit					
Linear	5.50	**0.967**	**0.091**	0.0175	0.979
Shapley	5.50	0.966	0.117	**0.0172**	**0.979**
6.0-bit					
Linear	6.00	0.969	0.110	0.0151	0.982
Shapley	6.00	**0.970**	**0.091**	**0.0145**	**0.983**
6.5-bit					
Linear	6.50	0.974	0.124	0.0116	0.986
Shapley	6.50	**0.983**	**0.059**	**0.0107**	**0.987**
7.0-bit					
Linear	6.98	**0.986**	**0.043**	0.0084	0.990
Shapley	6.98	0.980	0.059	**0.0079**	**0.990**

Table 21: INT Asymmetric Quantization Results (4-5-6-7-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	4.16	0.938	0.148	0.0277	0.970
Uniform	5.16	0.964	0.083	0.0068	0.992
Uniform	6.16	0.988	0.029	0.0017	0.998
Uniform	7.16	0.989	0.023	0.0004	1.000
Uniform	8.16	0.992	0.021	0.0001	1.000
4.5-bit					
Linear	4.50	**0.964**	**0.074**	**0.0171**	0.980
Shapley	4.50	0.960	0.115	0.0177	**0.982**
5.5-bit					
Linear	5.50	0.988	0.045	**0.0041**	**0.996**
Shapley	5.50	**0.991**	**0.027**	0.0047	0.995
6.0-bit					
Linear	6.00	0.989	**0.027**	**0.0025**	0.997
Shapley	6.00	**0.991**	0.029	0.0026	**0.997**
6.5-bit					
Linear	6.50	**0.995**	**0.023**	**0.0010**	**0.999**
Shapley	6.50	0.994	0.026	0.0011	0.999
7.0-bit					
Linear	7.00	**0.994**	**0.021**	**0.0006**	**1.000**
Shapley	7.00	**0.994**	**0.021**	0.0006	**1.000**

Table 22: INT Symmetric Quantization Results (4-5-6-7-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	4.12	0.938	0.145	0.0344	0.957
Uniform	5.12	0.964	0.083	0.0087	0.991
Uniform	6.12	0.985	0.051	0.0022	0.997
Uniform	7.12	0.989	0.023	0.0006	0.999
Uniform	8.12	0.997	0.005	0.0001	1.000
4.5-bit					
Linear	4.50	**0.963**	**0.112**	**0.0205**	0.974
Shapley	4.50	0.952	0.119	0.0212	**0.975**
5.5-bit					
Linear	5.50	0.983	**0.059**	**0.0051**	**0.994**
Shapley	5.50	**0.986**	0.091	0.0054	0.993
6.0-bit					
Linear	6.00	0.983	0.043	**0.0030**	**0.997**
Shapley	6.00	**0.986**	**0.025**	0.0031	0.996
6.5-bit					
Linear	6.50	**0.992**	**0.023**	**0.0013**	0.999
Shapley	6.50	0.989	0.025	0.0013	**0.999**
7.0-bit					
Linear	7.00	**0.992**	0.015	**0.0007**	0.999
Shapley	7.00	**0.992**	**0.005**	0.0007	**1.000**

Table 23: INT Asymmetric Distribution-Lossless (DL) Quantization Results

Config	Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baseline						
–	Uniform	4.16	0.938	0.148	0.0277	0.970
Lossless Constraint-Based						
4-8	Shapley	7.74	0.981	0.027	0.0020	0.997
4-5-6-7-8	Shapley	5.59	**0.994**	**0.020**	0.0039	0.996
4-5-6-7-8	Shapley	6.19	0.989	0.024	**0.0018**	**0.998**

Table 24: FP Asymmetric Quantization Results (4-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	5.00	0.955	0.130	0.0264	0.970
Uniform	8.00	0.989	0.024	0.0016	0.997
5.5-bit					
Linear	5.50	**0.972**	**0.066**	0.0168	0.980
Shapley	5.50	0.969	0.088	**0.0168**	**0.981**
6.0-bit					
Linear	6.00	**0.977**	**0.051**	**0.0139**	**0.984**
Shapley	6.00	0.975	0.088	0.0140	0.983
6.5-bit					
Linear	6.49	**0.980**	**0.042**	**0.0115**	0.987
Shapley	6.50	0.977	0.051	0.0116	**0.987**
7.0-bit					
Linear	6.99	0.981	0.042	**0.0083**	0.990
Shapley	6.99	**0.985**	**0.029**	0.0084	**0.990**

Table 25: FP Symmetric Quantization Results (4-8 Config)

Method	BW	EAR ↑	Margin ↓	KL ↓	PPL ↑
Uniform Baselines					
Uniform	8.00	0.989	0.024	0.0016	0.997
5.5-bit					
Linear	5.50	0.949	0.083	0.0163	0.983
Shapley	5.50	**0.957**	**0.076**	**0.0161**	**0.983**
6.0-bit					
Linear	6.00	0.960	**0.076**	0.0136	0.984
Shapley	6.00	**0.971**	0.083	**0.0134**	**0.985**
6.5-bit					
Linear	6.50	0.966	**0.074**	**0.0107**	0.987
Shapley	6.50	**0.969**	0.080	0.0109	**0.987**
7.0-bit					
Linear	7.00	**0.974**	0.076	0.0077	0.990
Shapley	7.00	0.969	**0.074**	**0.0076**	**0.991**