QUASAR: LOWERING THE LOSS FLOOR OF QUANTIZATION-AWARE TRAINING WITH LOSS-AWARE RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language model inference shifts toward lower precision, post-training quantization (PTQ) becomes increasingly brittle, making quantization-aware training (QAT) essential for preserving model quality. However, QAT has a structural mismatch: gradient updates are applied to latent full-precision weights, while the loss and gradients are computed on lossy reconstructions of those weights. This mismatch can lead to suboptimal training trajectories and a higher loss floor. Second-order PTQ methods address a similar problem by minimizing loss-aware reconstruction error, but applying such expensive reconstruction repeatedly during QAT as the weights evolve is impractical. We introduce QUASAR, a QAT method that brings lightweight, loss-aware reconstruction into the training loop. At each training step, QUASAR reconstructs the latent weights by searching over a small set of clipping ranges and fitting dequantization parameters through saliency-weighted least squares. We use an exponential moving average of squared gradients as the per-parameter saliency signal. Our theoretical analysis shows that optimizing QUASAR's reconstruction objective tightens both the convergence and final-loss bounds of QAT. We evaluate QUASAR across four model families and across INT4, INT3, INT2, and NVFP4 quantization formats. QUASAR consistently achieves lower training and evaluation loss than competitive QAT methods and outperforms QAT and PTQ baselines on downstream benchmarks. At INT2, QUASAR improves average accuracy over the best QAT baseline by 13.3 points with quantization-aware distillation and by 10.9 points with QAT on mathematical reasoning data. Notably, after distillation on only $\sim$600M tokens, QUASAR's INT4 Gemma-4 E4B checkpoint outperforms the corresponding QAT checkpoint released by Google, with 66% lower KL divergence and 1.8 points higher average accuracy.

1 INTRODUCTION

As large language models (LLMs) grow in size and adoption, inference is overtaking training as the dominant recurring cost of deployment (Desislavov et al., 2023; Sardana et al., 2024). To reduce this cost, inference is rapidly moving toward low-precision formats such as FP4 and INT4 (Rouhani et al., 2023; NVIDIA et al., 2025; Frantar et al., 2023), which shrink the model's memory footprint, reduce decoding latency, and enable higher concurrency and throughput (Lin et al., 2025). However, since models are typically trained in high precision but served in low precision (Gholami et al., 2021), preserving model quality as precision becomes increasingly aggressive remains a critical challenge. Post-training quantization (PTQ) has been the standard way to convert a trained model to low precision, but it is becoming increasingly brittle on newer models. Reasoning and agentic models often operate over long contexts, where quantization errors can accumulate and degrade performance (Liu et al., 2025a; Li et al., 2025b; Mekala et al., 2025). PTQ models may also exhibit behavioral shifts and biases relative to their full-precision counterparts (Marchisio et al., 2024; Marcuzzi et al., 2026; Dutta et al., 2024). Producing a native low-precision model with stronger quality guarantees therefore requires training under the same precision used for serving.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

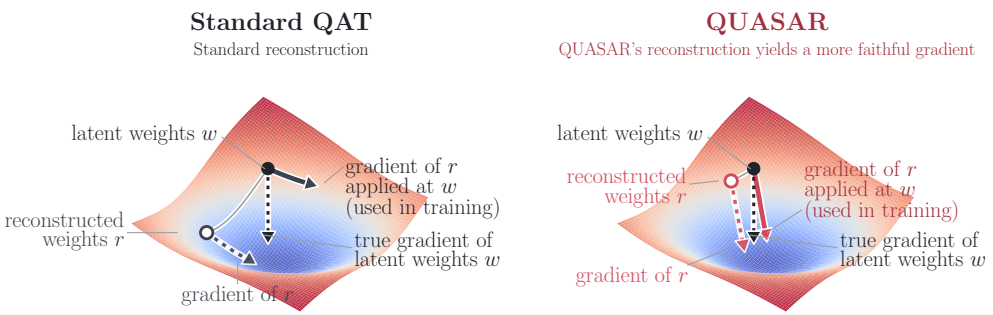


Figure 1: **The structural mismatch of gradients in standard QAT versus QUASAR.** Under the STE, both methods evaluate the gradient at the reconstructed weights r and apply it to the latent weights w. QUASAR minimizes the loss-aware reconstruction error, making the gradient at r a more faithful proxy for the gradient at w.

Quantization-aware training (QAT) addresses this need by inserting quantization into the training loop, allowing the model to adapt to quantization noise as it learns. QAT, however, introduces a structural mismatch into optimization (Figure 1): the forward pass and loss computation use quantized–dequantized reconstruction weights r, while the gradient updates are applied to the latent full-precision weights w. Because quantization has zero gradient almost everywhere, QAT typically uses the straight-through estimator (STE) (Bengio et al., 2013), approximating $\frac{\partial \mathcal{L}}{\partial w}$ with $\frac{\partial \mathcal{L}}{\partial r}$. This surrogate is generally not the optimal descent direction for the latent weights, leading training along a suboptimal trajectory. The consequence is a *loss-floor gap*: given the same model and training data, QAT converges to a higher final training and evaluation loss than full-precision training (Chen et al., 2025b; Kumar et al., 2025). Prior QAT methods narrow this gap by softening the rounding operation, learning quantization parameters jointly with the weights, or improving initialization (Gong et al., 2019; Esser et al., 2020; Du et al., 2024). We instead draw on a complementary insight from second-order PTQ methods: the excess training loss caused by QAT's structural mismatch can be reduced by optimizing reconstruction.

A second-order expansion of the loss around the full-precision weights w, assuming that the first-order term is negligible near an optimum, gives

$$\mathcal{L}(r) - \mathcal{L}(w) \approx \frac{1}{2} S, \qquad S = (r - w)^\top H (r - w),$$

where $H(w)$ is the Hessian evaluated at w. We refer to S as the *loss-aware reconstruction error*. Many modern PTQ methods (Nagel et al., 2020; Li et al., 2021; Frantar et al., 2022; 2023; Chee et al., 2023) minimize this quantity, or an approximation to it, to construct a low-precision model whose loss remains close to that of a frozen full-precision model. Applying the same principle to QAT would keep the low-precision loss $\mathcal{L}(r)$ closer to the full-precision loss $\mathcal{L}(w)$ as the latent weights evolve, thereby mitigating the effect of the structural mismatch. The challenge, however, is computational cost. In PTQ, the full-precision model is frozen, so its reconstruction needs to be optimized only once, typically requiring minutes to hours for a billion-parameter LLM. In QAT, however, the latent weights change at every step, requiring the reconstruction to be continually re-optimized. Running a conventional second-order optimization procedure on every forward pass would therefore be prohibitively expensive.

We introduce **QUASAR** (**Qu**antization-**a**ware training with lo**ss**-**a**ware **r**econstruction), a QAT method that continually minimizes loss-aware reconstruction error to improve the training trajectory and lower the loss floor. QUASAR decomposes reconstruction into two stages: *quantization*, which maps full-precision weights to discrete codes while treating the clipping range as a free parameter, and *dequantization*, which maps those codes back to reconstructed weights using a learned scale and offset. To make the loss-aware objective tractable during training, QUASAR approximates the Hessian using an exponential moving average of squared gradients, yielding per-parameter saliency scores that estimate each weight's effect on the loss. It then searches over a small set of candidate clipping ranges, each defining a different assignment of discrete codes; for each assignment, it solves for the dequantization parameters in closed form by saliency-weighted least squares and selects the candidate with the smallest loss-aware reconstruction error. Our theoretical analysis shows that reducing this error tightens the QAT convergence bound and final-loss bound (Section 4).

Our contributions are threefold. **(i)** We identify loss-aware reconstruction error, previously optimized for frozen models in second-order PTQ, as a principled objective for QAT. We introduce QUASAR to minimize this objective at every training step using an exponential moving average of squared gradients, closed-form saliency-weighted least squares, and a small clipping-range search. **(ii)** Under empirically validated assumptions, we prove that reducing this error tightens bounds on both QAT convergence and the loss of the final quantized model. **(iii)** Across four model families and four quantization formats, QUASAR consistently reduces loss and improves downstream performance over competitive QAT and PTQ baselines. At INT2, it improves average accuracy by up to 13.3 points over the strongest QAT baseline; at INT4, its Gemma-4 E4B checkpoint achieves 66% lower held-out KL and 1.8 points higher average accuracy than Google's public QAT checkpoint.

2 Related Work

QUASAR sits between two lines of work: PTQ and QAT. PTQ treats quantization as a one-shot algorithm: given a trained model and a small calibration set, select low-precision values that damage the loss least, typically by minimizing a second-order expansion of the loss (Hassibi & Stork, 1992; Frantar et al., 2022; 2023) or learning the rounding against a reconstruction objective (Nagel et al., 2020; Li et al., 2021; Tseng et al., 2026). All rest on the premise that the weights are frozen, so the reconstruction happens only once. A second PTQ line spreads outliers with random orthogonal transforms and replaces the quantization grid with lattice, trellis, or additive codebooks (Chee et al., 2023; Tseng et al., 2024a;b; Egiazarian et al., 2024); these are the strongest PTQ methods at two and three bits, but their inference-time transforms and decodes hurt throughput, while hardware moves toward simpler formats such as MXFP4 and NVFP4 (Rouhani et al., 2023; NVIDIA et al., 2025).

Below four bits, the PTQ methods that do not add inference overhead degrade sharply. QAT reaches higher quality at these bit widths by learning step sizes (Esser et al., 2020), supervising with the full-precision model (Liu et al., 2023; Du et al., 2024), or freezing the quantized weights and training low-rank adapters (Dettmers et al., 2023); none of these optimize the map from latent weights to reconstructed weights, which is set by heuristics. QUASAR directly optimizes that map: it brings the reconstruction PTQ solves once into the training loop as the weights adapt, while still deploying exactly as an RTN-quantized model. Other methods are complementary and compose with QUASAR: EfficientQAT's second stage learns only the step sizes, the same quantizer parameter as LSQ, and its block-wise first stage is a complementary initialization (Chen et al., 2025a); ParetoQ splits a fixed training budget between full-precision training and QAT (Liu et al., 2025b).

3 Methodology

We first describe weight reconstruction in standard QAT and explain how reconstruction error affects the training loss. We then introduce QUASAR, which makes this reconstruction loss-aware through an efficient curvature approximation, scale search over clipping ranges, and optimal dequantization.

3.1 Preliminary: Reconstruction in QAT

QAT maintains and updates full-precision latent weights w, but computes the forward pass and loss using reconstructed weights r. At each step, QAT maps w to low-bit codes q and then dequantizes those codes to obtain r: $w \mapsto q \mapsto r$. We refer to this process as *weight reconstruction*. It exposes the model to inference-time quantization error during training while allowing the latent weights to adapt. Because the code assignment involves rounding and has no useful derivative, QAT uses the straight-through estimator (STE) (Bengio et al., 2013): $\frac{\partial \mathcal{L}}{\partial w} \approx \frac{\partial \mathcal{L}}{\partial r}$. The gradient evaluated at r is therefore applied to w (Figure 1).

For asymmetric integer quantization, each weight matrix is partitioned into groups of g contiguous weights, commonly with $g = 32$ or 128. A group $w \in \mathbb{R}^g$ is represented by b-bit integer codes $q_i \in \{0, \dots, 2^b - 1\}$. All weights in the group share a scale s and an offset z, which determine the quantization grid. Quantization assigns codes and dequantization reconstructs weights as

$$q_i = \mathrm{clamp}_{[0,2^b-1]}\left(\left\lfloor \frac{w_i - z}{s} \right\rceil\right), \qquad r_i = sq_i + z. \tag{1}$$

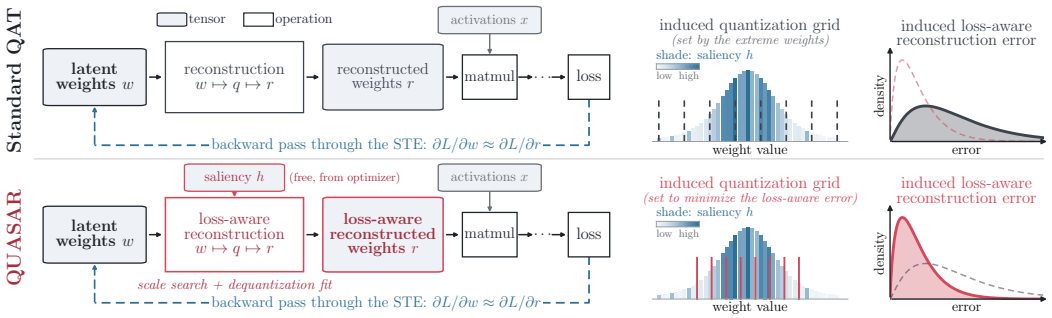


Figure 2: The training loops for standard QAT and QUASAR differ only in the reconstruction $w \mapsto q \mapsto r$. Standard QAT determines the quantization grid from each group's extreme weights; QUASAR searches over code assignments and fits the dequantizer to minimize loss-aware reconstruction error. The right panels show the resulting grids and error distributions; empirical measurements appear in Appendix A.

Reconstruction has two stages: *code assignment*, $w \mapsto q$, and *dequantization*, $q \mapsto r$. Standard QAT uses the minimum and maximum weights in each group for both stages, setting $z = \min_i w_i$ and $s = (\max_i w_i - \min_i w_i)/(2^b - 1)$. This uniform grid represents the full range of weight values without clipping and preserves the extreme values in the group, but it may not be optimal for reducing the training loss.

Reconstruction quality directly affects the training loss of QAT. At step t, consider the difference between the loss at the reconstructed weights and the loss at the latent weights, $\mathcal{L}(r_t) - \mathcal{L}(w_t)$. When the first-order term is negligible, as near a stationary region, a second-order expansion around w_t, with $H_t = \nabla^2 \mathcal{L}(w_t)$, gives

$$\mathcal{L}(r_t) - \mathcal{L}(w_t) \approx \frac{1}{2} S_t, \qquad \text{where } S_t = (r_t - w_t)^\top H_t (r_t - w_t). \tag{2}$$

We call S_t the *loss-aware reconstruction error*. It measures the local change in loss caused by reconstructing the latent weights. Because both the forward pass and the STE gradient are evaluated at r_t, this error can affect every optimization step. Second-order PTQ methods minimize this objective, or a related approximation, once for a frozen model (Nagel et al., 2020; Frantar et al., 2023; Tseng et al., 2026), but the process can take hours for a large model. Repeating this optimization at every QAT training step is impractical because the latent weights change after every update. QUASAR instead minimizes a close approximation to S_t at every training step using a lightweight procedure that combines scale search with a closed-form fit for dequantization.

3.2 A Tractable Loss-Aware Objective

QUASAR aims to minimize the loss-aware reconstruction error S_t in Equation (2). The exact Hessian H in S_t is intractable to compute, so an approximation is required. One could consider computing the Hessian Vector Product (HVP), but it is also too expensive since it requires computing an HVP for every candidate reconstruction for every training step. We therefore use the diagonal Fisher information matrix as a proxy for the Hessian. Following Li et al. (2025a), we approximate it with an exponential moving average (EMA) of squared gradients. Adam and AdamW already maintain this quantity as the second-moment estimate v_t (Kingma & Ba, 2015), allowing QUASAR to reuse it without additional memory overhead. The EMA also smooths noise from individual training steps.

Let $g_i = \partial \mathcal{L}/\partial w_i$ be the gradient with respect to the latent weight w_i, and define $h_i = \mathrm{EMA}(g_i^2)$. We use h_i as the ith diagonal entry of the Hessian approximation, giving the objective

$$S_t \;\approx\; \widehat{S}_t = \sum_i h_i \, (r_i - w_i)^2. \tag{3}$$

We call h_i the *saliency* of w_i. Saliency varies substantially within a weight group (Appendix A, Figure 8). Because all g weights in a group share dequantization parameters, reducing the reconstruction error of one weight can increase that of another. The weights h_i govern this tradeoff: QUASAR prioritizes accurate reconstruction of high-saliency weights while tolerating more error on low-saliency weights (Appendix A, Figure 5).

Algorithm 1 QUASAR reconstruction of one weight group, run at every training step. A fully specified version appears in Algorithm 2 (Appendix A.1).

Require: latent weights w of one group; saliency h (Adam's second moment); clipping factors $\mathcal{F} \subseteq (0, 1]$

$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $q_f \leftarrow$ quantized codes of w using the clipping range I_f ▷ Eqs. (1) and (4)
  $(s_f, z_f) \leftarrow h$-weighted least-squares fit of (s, z) given (w, q_f) ▷ Eq. (6)
  $r_f \leftarrow s_f q_f + z_f \mathbf{1}$ ▷ best reconstruction for clipping factor f
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f; r \leftarrow r_f$ **end if**
end for
return r

3.3 Optimizing Both Stages of Reconstruction

To minimize the objective in Equation (3), QUASAR optimizes code assignment $w \mapsto q$ and dequantization $q \mapsto r$ separately. Standard QAT uses the same scale and offset to assign codes and reconstruct weights. In contrast, QUASAR searches over candidate clipping scales to find code assignments and, for each assignment, computes the optimal dequantization scale and offset in closed form. It then selects the candidate with the smallest loss-aware reconstruction error. Algorithm 1 summarizes the procedure.

A weighted least-squares fit gives the optimal dequantization. For a fixed code assignment q, QUASAR solves for (s, z) in $r_i = sq_i + z$ to minimize Equation (3). This is a weighted linear regression and therefore has a closed-form solution (Equation (6), Appendix A.1).

Scale search finds good code assignments. The closed-form fit is optimal for a fixed q, but the code assignment itself may be far from optimal. QUASAR therefore searches over scale factors that shrink the clipping range. Each range creates a different code assignment. For a group with midpoint c and half-range ρ, a factor $f \in (0, 1]$ defines the assignment range I_f, where

$$c = \frac{\min_i w_i + \max_i w_i}{2}, \qquad \rho = \frac{\max_i w_i - \min_i w_i}{2}, \qquad I_f = [c - f\rho,\, c + f\rho]. \tag{4}$$

The case $f = 1$ is the full min–max range used by standard QAT. A factor $f < 1$ narrows the range and clips weights outside it, so the quantization levels cover a smaller interval with finer spacing. In our integer experiments, we search $\mathcal{F} = \{0.30, 0.35, \ldots, 1.00\}$.

For each candidate $f \in \mathcal{F}$, QUASAR first uses the corresponding clipping range to assign codes q_f. It then fits the optimal dequantization parameters (s_f, z_f) for q_f using Equation (6) and computes the resulting error $\widehat{S}_t(f)$. After evaluating all candidates, QUASAR uses the reconstruction associated with $f^\star = \arg\min_{f \in \mathcal{F}} \widehat{S}_t(f)$. This search is performed as part of each QAT forward pass. Its empirical behavior is shown in Figures 6 and 7 in Appendix A.

QUASAR changes only the weight reconstruction used in the forward pass. It leaves activation quantization and backpropagation unchanged: gradients still pass through the weight quantizer with the standard STE, and neither scale search nor dequantization fitting requires a separate backward rule. The same two-stage procedure applies to asymmetric and symmetric integer formats as well as floating-point formats such as NVFP4 and MXFP4 (the corresponding algorithms are presented in Appendix A.1). We evaluate QUASAR for NVFP4 in Section 5.2.

4 Theory: Reconstruction Error Bounds QAT Final Loss

In this section, we show that QUASAR lowers the excess loss of the final quantized model, $\mathcal{L}(r_T) - \mathcal{L}^\star$. This gap is bounded by QUASAR's loss-aware reconstruction error, so minimizing that error is a natural target. Under the identity STE, the gradient evaluated at the reconstruction r_t is applied to the latent weights w_t, creating the gradient mismatch $e_t := \nabla\mathcal{L}(r_t) - \nabla\mathcal{L}(w_t)$. Prior work has studied when STE gradients remain useful descent directions (Yin et al., 2019), how curvature shapes QAT plateaus (Li et al., 2026), and how a systematic gradient bias slows SGD (Ajalloeian & Stich, 2020). To the best of our knowledge, no prior work has studied the relationship between the reconstruction error of the quantizer to the convergence of QAT or to the loss of the final quantized model. Let

$\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be the training objective with $\mathcal{L}^\star := \inf_x \mathcal{L}(x)$, and $T \geq 1$ be the number of training steps. At step t, $w_t, r_t \in \mathbb{R}^d$ are the latent and reconstructed weights, $h_t \in \mathbb{R}^d_+$ the saliencies, and $\widehat{S}_t := \sum_i h_{t,i}(r_{t,i} - w_{t,i})^2$ the objective of Section 3 at step t. $\mathbb{E}_t[\cdot]$ denotes expectation conditional on the training history $\mathcal{H}_t$, which determines h_t and r_t.

Assumptions. We make four assumptions. **(1)** The objective $\mathcal{L}$ is well behaved: twice continuously differentiable, L-smooth, and bounded below. **(2)** The STE gradient is the true gradient at the reconstruction plus zero-mean noise: $G_t = \nabla\mathcal{L}(r_t) + \xi_t$ with $\mathbb{E}_t[\xi_t] = 0$ and $\mathbb{E}_t\|\xi_t\|^2 \leq M\|\nabla\mathcal{L}(r_t)\|^2 + \sigma^2$ for constants $M, \sigma^2 \geq 0$, the biased-SGD noise condition (Ajalloeian & Stich, 2020); the update is $w_{t+1} = w_t - \eta G_t$. **(3)** The reconstruction error controls the gradient mismatch: for some finite constant C and every $t \in \{0, \ldots, T\}$, $\|\nabla\mathcal{L}(r_t) - \nabla\mathcal{L}(w_t)\|^2 \leq C\,\widehat{S}_t$. Appendix B.1 gives a sufficient curvature condition with $C = L^2/h_{\min}$ under a saliency floor $h_{\min} > 0$, Figure 14 (Appendix C) confirms the inequality empirically at all tested bit widths, and Appendix B.2 discusses the assumptions further. Appendix A.2 compares the H- and H^2-motivated saliency weightings. **(4)** The objective satisfies the global Polyak–Łojasiewicz (PL) inequality $\|\nabla\mathcal{L}(x)\|^2 \geq 2\mu(\mathcal{L}(x) - \mathcal{L}^\star)$ for all $x \in \mathbb{R}^d$ and some $\mu > 0$ (Polyak, 1963; Karimi et al., 2016); it is needed only for the final-model bound.

Theorem 1 (Reconstruction error controls QAT convergence and final loss)**.** *Under Assumptions 1–3, let* $0 < \eta < \frac{1}{L(1+M)}$. *The following statements hold.*

(i) ***Convergence.***

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\nabla\mathcal{L}(w_t)\|^2 + \big(1 - L\eta(1+M)\big)\|\nabla\mathcal{L}(r_t)\|^2\big] \leq \frac{2(\mathcal{L}(w_0) - \mathcal{L}^\star)}{\eta T} + L\eta\sigma^2 + \frac{C}{T}\sum_{t=0}^{T-1}\mathbb{E}\widehat{S}_t.$$

(ii) ***Final quantized model.*** *If Assumption 4 also holds and* $\eta < \frac{1}{\mu}$*, then, for a random training iterate* τ *(Appendix B.1),*

$$\mathbb{E}[\mathcal{L}(r_T) - \mathcal{L}^\star] \leq \frac{2L}{\mu}(1 - \eta\mu)^T(\mathcal{L}(w_0) - \mathcal{L}^\star) + \frac{LC}{\mu^2}\mathbb{E}\widehat{S}_\tau + \frac{L^2\eta\sigma^2}{\mu^2} + \frac{C}{\mu}\mathbb{E}\widehat{S}_T. \quad (5)$$

Proofs are presented in Appendix B. Theorem 1 shows that the QAT convergence bound contains the usual stochastic-optimization terms (initialization, minibatch noise) plus a quantization term given by the loss-aware reconstruction error. QUASAR directly minimizes this term at every step, hence tightening the bound on the quantized model's final loss. Appendix C supports the theorem's assumptions (Figures 14 and 16), evaluates the bound (Figure 17), shows that reconstruction error tracks final loss (Figures 10, 12, and 13), and repeats the comparison under SGD (Figure 18).

5 Experiments

In this section, we evaluate QUASAR against competitive QAT and PTQ baselines. We assess its ability to preserve quantized model quality using Quantization-aware Distillation (QAD), to adapt quantized models to learn new tasks using QAT, and to generalize to different quantization formats including INT4/INT3/INT2 and NVFP4. Moreover, we compare QUASAR QAT'ed checkpoints against the QAT checkpoints released by Google (Gemma Team, 2026) under the same format.

Setup and baselines. For healing, we apply QAD to Qwen3-4B-Thinking-2507 (Yang et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) at INT2, INT3, and INT4 using OpenPerfectBlend data (Labonne, 2024) and logits from the full-precision teachers. For adaptation, we train all weights of Qwen3-4B-Base on OpenMathReasoning (Moshkov et al., 2025) using cross-entropy. We also apply QAD in NVFP4 to Qwen3-8B (Yang et al., 2025), Qwen3.5-9B (Qwen Team, 2026b), and Muse-Glimmer-30B (Meta Superintelligence Lab, 2026), and evaluate QUASAR against Google's QAT checkpoint, Gemma 4 (Gemma Team, 2026), and other independent checkpoints. Baselines in each experiment use the same data, objective, training budget, and tuned learning rate, with matched inference formats. We compare against Standard QAT (Section 3.1), LSQ (Esser et al., 2020), Denoising QAT (Ye et al., 2026), BitDistiller (Du et al., 2024), RTN, GPTQ (Frantar et al., 2023), and AWQ (Lin et al., 2024); for GGUF, we also include imatrix PTQ, which uses calibration activations to estimate weight importance. Full training protocols and hyperparameters are in Appendix D.

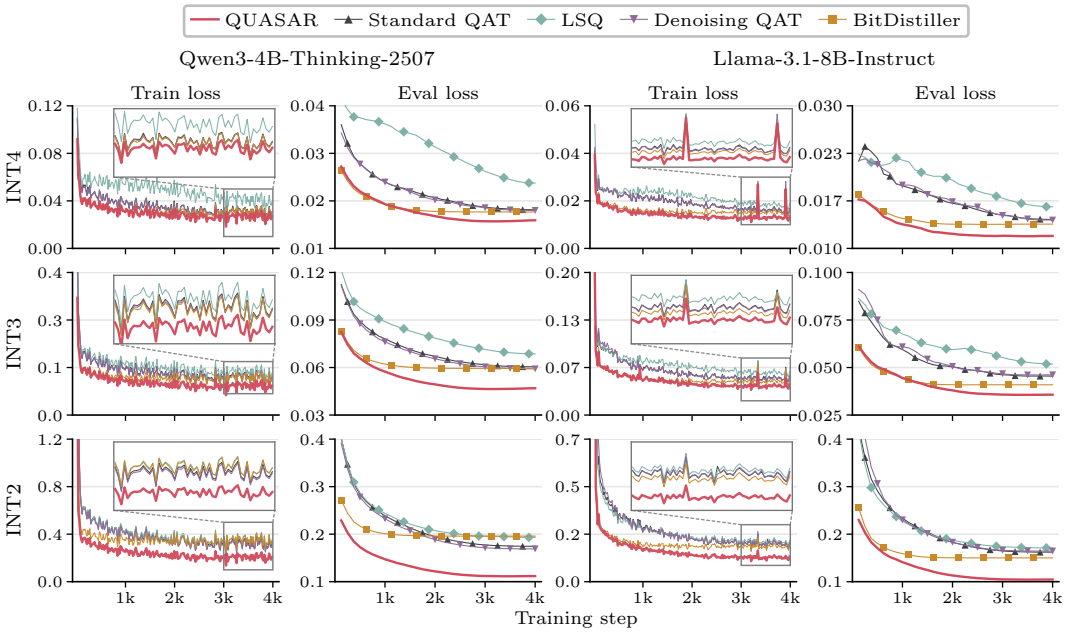


Figure 3: QAD of Qwen3-4B-Thinking-2507 (left) and Llama-3.1-8B-Instruct (right) at INT4, INT3, and INT2 on Open-PerfectBlend. We show training loss over the full run, the final 1,000 steps, and held-out evaluation loss. All losses are forward KL to the full-precision model.

Evaluation. We evaluate quantized checkpoints and report fidelity to the full-precision model together with downstream accuracy. Our benchmark suites cover standard language understanding (MMLU, ARC, HellaSwag, WinoGrande, TruthfulQA, and IFEval) (Hendrycks et al., 2021a; Clark et al., 2018; Zellers et al., 2019; Sakaguchi et al., 2020; Lin et al., 2022; Zhou et al., 2023); mathematical and scientific reasoning (MATH-500, GSM8K, AIME, HMMT, MMLU-Pro, SuperGPQA, and GPQA-Diamond) (Hendrycks et al., 2021b; Lightman et al., 2023; Cobbe et al., 2021; Mathematical Association of America; HMMT; Wang et al., 2024; M-A-P Team et al., 2025; Rein et al., 2023); coding (LiveCodeBench and OJBench) (Jain et al., 2025; Wang et al., 2025); and long-context performance (LongBench-v2, RULER, MRCR, and LongProc) (Bai et al., 2024; Hsieh et al., 2024; OpenAI, 2025; Ye et al., 2025).

5.1 Results on Healing and Adaptation

Healing with QAD. Figure 3 presents the training and evaluation loss of QUASAR and baseline QAT methods, and Table 1 presents the downstream accuracy of the final quantized checkpoints of QAT and PTQ. Across both models and all bit widths, QUASAR reaches the lowest training and evaluation loss (KL). At INT2, it reduces KL by about 30% over the strongest QAT baseline. On reasoning, coding, and long-context tasks, QUASAR improves average accuracy by 13.3 points for Qwen and 2.2 points for Llama (Table 1). Additional results and analyses are in Appendix E.

Adaptation with QAT. Table 2 presents the downstream accuracy of INT4/INT3/INT2 Qwen3-4B-Base adapted using different QAT methods, and Figure 30 in the Appendix presents the training and evaluation loss curves. QUASAR has the lowest held-out perplexity and the best or tied-best average accuracy at every bit width. At INT2, QUASAR reaches 29.6 average accuracy, 10.9 points above the strongest QAT baseline and 29.0 points above full-precision fine-tuning followed by PTQ (Table 2). It is also the only INT2 quantized model to outperform the original base model and score above zero on HMMT'25. Additional results are in Appendix F.

5.2 NVFP4 and Checkpoint Comparisons

NVFP4 QAD. Table 3 reports downstream accuracy for Qwen3-8B, Qwen3.5-9B, and Muse-Glimmer-30B in NVFP4, all trained with the same QAD healing recipe as in Section 5.1. Figure 34 in Appendix G shows the corresponding training and evaluation loss curves for the two Qwen mod-

Table 1: Reasoning, coding, and long-context results for Qwen3-4B-Thinking-2507 and Llama-3.1-8B-Instruct at INT2, INT3, and INT4. We report KL and top-1 agreement between quantized checkpoints and the BF16 original, as well as benchmark accuracy.

	Qwen3-4B-Thinking-2507											Llama-3.1-8B-Instruct							
Method	KL ↓	Top-1 ↑	HMMT '26	AIME '25	MATH-500	MMLU-Pro	SuperGPQA	LCB v6	LongBench-v2	RULER	**Avg.**	KL ↓	Top-1 ↑	MATH-500	MMLU-Pro	SuperGPQA	LCB v6	LongBench-v2	RULER	**Avg.**
Teacher (BF16)	–	–	43.1	77.9	98.2	72.7	48.1	74.6	44.5	87.9	68.4	–	–	49.6	42.3	22.4	17.0	30.0	89.6	41.8
2-bit (INT2)																				
RTN	11.166	0.5	0.0	0.0	1.9	0.0	10.0	0.0	0.0	0.0	1.5	10.810	0.5	0.0	0.0	10.2	0.0	0.0	0.0	1.7
GPTQ	1.140	67.7	0.0	0.0	2.6	3.2	8.4	0.0	0.0	0.3	1.8	2.380	51.8	2.9	1.8	8.7	0.0	0.0	0.1	2.3
AWQ	3.016	37.0	0.0	0.0	2.4	0.0	5.2	0.0	0.0	0.0	1.0	6.769	12.9	1.6	0.0	10.7	0.0	0.0	0.0	2.0
Standard QAT	0.186	89.9	1.0	1.7	49.5	26.6	17.4	8.2	20.5	39.3	20.5	0.163	90.1	21.6	16.8	11.8	4.0	15.1	5.5	12.5
LSQ	0.205	89.2	0.8	0.5	47.2	26.2	17.8	7.4	20.7	41.7	20.3	0.172	89.6	19.9	15.7	11.6	2.3	15.3	52.9	19.6
Denoising QAT	0.179	90.1	1.0	1.4	51.8	27.4	19.0	9.2	18.9	39.4	21.0	0.165	90.1	20.8	16.5	11.0	3.6	8.5	0.0	10.1
BitDistiller	0.209	89.1	0.6	1.1	51.4	29.3	16.8	9.4	18.7	44.9	21.5	0.151	90.2	21.7	19.0	11.8	5.5	**20.5**	63.5	23.7
QUASAR (ours)	**0.126**	**92.0**	**6.2**	**14.4**	**77.4**	**45.8**	**25.4**	**23.4**	**26.6**	**59.6**	**34.8**	**0.105**	**92.0**	**30.0**	**22.6**	**13.7**	**7.0**	15.9	**66.4**	**25.9**
3-bit (INT3)																				
RTN	0.597	78.8	0.0	0.7	13.0	16.3	15.6	0.0	8.7	49.3	13.0	0.276	85.6	8.2	17.2	12.5	0.7	21.1	49.0	18.1
GPTQ	0.130	91.5	31.2	49.4	**95.8**	63.7	39.5	49.4	31.0	73.3	54.1	0.076	92.8	38.6	31.9	18.4	10.4	27.6	80.9	34.6
AWQ	0.240	88.1	22.7	31.0	90.6	61.5	37.0	35.7	30.2	76.0	48.1	0.197	88.0	16.2	15.3	13.9	2.4	22.7	69.0	23.3
Standard QAT	0.062	94.2	29.9	47.4	94.2	64.9	40.0	53.3	29.6	80.8	55.0	0.046	94.7	39.5	34.8	18.6	12.7	30.2	81.7	36.3
LSQ	0.070	93.8	28.3	45.5	93.3	64.1	38.2	52.2	32.8	80.5	54.4	0.052	94.2	39.3	35.0	17.1	13.7	27.2	**86.1**	36.4
Denoising QAT	0.060	94.3	29.3	49.3	94.7	64.5	39.7	52.3	32.8	79.0	55.2	0.046	94.7	41.2	34.6	18.2	13.0	29.2	80.3	36.1
BitDistiller	0.065	94.2	29.7	52.5	95.1	65.8	40.6	57.7	33.0	83.5	57.3	0.041	94.8	42.5	35.9	19.3	13.1	**31.2**	84.4	37.7
QUASAR (ours)	**0.054**	**94.9**	**31.7**	**60.5**	95.2	**66.7**	**41.5**	**60.2**	**35.8**	**84.4**	**59.5**	**0.036**	**95.2**	**43.4**	**36.4**	**19.7**	**16.2**	26.0	85.8	**37.9**
4-bit (INT4)																				
RTN	0.108	92.2	35.7	62.1	96.8	69.7	44.4	65.2	38.8	83.3	62.0	0.044	94.4	44.5	35.5	20.1	14.2	29.6	**89.0**	38.8
GPTQ	0.027	96.2	38.8	**74.8**	**97.8**	71.4	**46.5**	70.6	38.2	87.4	65.7	0.015	96.8	47.9	40.0	20.6	**16.5**	**31.2**	88.5	40.8
AWQ	0.050	94.7	36.6	70.6	97.6	71.6	45.7	70.8	39.0	86.8	64.8	0.032	95.2	44.8	39.8	21.3	13.2	30.8	88.1	39.7
Standard QAT	0.019	96.9	36.8	72.1	97.7	71.4	**46.5**	70.9	39.0	87.1	65.2	0.014	97.0	47.7	42.1	21.1	15.8	31.0	**89.0**	**41.1**
LSQ	0.024	96.4	**39.3**	68.5	97.5	70.8	45.6	69.3	36.4	87.3	64.3	0.016	96.8	48.3	41.9	21.2	15.5	29.8	87.5	40.7
Denoising QAT	0.018	96.9	38.8	71.5	97.2	70.9	46.4	71.6	**42.9**	86.9	65.8	0.014	97.0	48.1	42.1	**21.7**	14.3	**31.2**	88.9	**41.1**
BitDistiller	0.018	96.9	38.5	73.9	97.3	71.4	**46.5**	71.1	40.4	86.8	65.7	0.013	97.0	**48.4**	41.0	21.0	15.7	31.0	88.9	41.0
QUASAR (ours)	**0.016**	**97.1**	38.5	73.8	97.5	**71.7**	46.2	**72.2**	41.0	**88.0**	**66.1**	**0.012**	**97.3**	47.6	**42.2**	20.8	16.4	30.0	**89.0**	41.0

Table 2: Comparison of QAT and PTQ methods for adapting Qwen3-4B-Base on OpenMathReasoning at INT2, INT3, and INT4. We report held-out perplexity and accuracy on five math benchmarks.

	2-bit (INT2)							3-bit (INT3)							4-bit (INT4)						
Method	PPL ↓	MATH-500	GSM8K	AIME'24	AIME'25	HMMT'25	**Avg.**	PPL ↓	MATH-500	GSM8K	AIME'24	AIME'25	HMMT'25	**Avg.**	PPL ↓	MATH-500	GSM8K	AIME'24	AIME'25	HMMT'25	**Avg.**
Base (no SFT)	–	41.0	66.4	9.6	3.3	0.8	24.2	–	41.0	66.4	9.6	3.3	0.8	24.2	–	41.0	66.4	9.6	3.3	0.8	24.2
FP-SFT	1.59	83.8	91.3	22.9	22.5	10.0	46.1	1.59	83.8	91.3	22.9	22.5	10.0	46.1	1.59	83.8	91.3	22.9	22.5	10.0	46.1
RTN	3.5×10^5	0.9	0.3	0.0	0.0	0.0	0.2	2.24	12.8	7.6	0.0	1.7	0.0	4.4	1.66	78.2	90.0	15.0	13.3	7.5	40.8
GPTQ	3.12	2.1	1.1	0.0	0.0	0.0	0.6	1.68	71.0	85.0	11.2	11.2	4.2	36.5	1.61	81.7	90.6	19.1	20.5	8.5	44.1
AWQ	130	1.5	1.2	0.0	0.0	0.0	0.5	1.83	57.3	65.1	10.0	8.3	1.7	28.5	1.63	80.3	90.5	15.8	17.5	8.3	42.5
Standard QAT	1.79	38.7	51.9	2.1	0.8	0.0	18.7	1.64	75.8	87.5	15.0	**16.2**	4.6	39.8	1.60	82.1	90.8	**21.2**	20.6	10.7	45.1
LSQ	1.82	36.5	49.9	0.8	0.4	0.0	17.5	1.65	73.7	87.0	12.5	11.7	3.8	37.7	1.62	79.0	89.1	17.9	17.1	6.7	42.0
Denoising QAT	1.78	39.2	52.5	0.8	0.8	0.0	18.7	1.64	75.9	87.3	14.2	13.3	**7.1**	39.6	1.60	82.1	90.4	20.0	**21.4**	9.9	44.8
BitDistiller	1.87	33.6	57.0	1.2	0.4	0.0	18.4	1.64	75.7	87.8	**16.2**	15.4	5.0	40.0	1.60	82.7	90.6	20.3	20.8	9.3	44.7
QUASAR (ours)	**1.71**	**60.7**	**75.4**	**4.0**	**6.5**	**1.5**	**29.6**	**1.62**	**78.5**	**88.6**	15.9	**16.2**	6.0	**41.1**	**1.60**	**82.8**	**90.9**	20.8	19.8	**11.2**	**45.1**

els. All checkpoints are evaluated using vLLM's native NVFP4 path (Kwon et al., 2023; vLLM Project, 2026b). For Qwen3-8B and Qwen3.5-9B, QUASAR reduces held-out KL divergence by 12% and 15% relative to Standard QAT, respectively. On the more challenging reasoning, coding, and long-horizon benchmarks, Qwen3.5-9B achieves an average score of 55.7, compared with 52.6 for Standard QAT and 39.4 for GPTQ (Table 3). These results also highlight the value of QAD: both QAD checkpoints remain stable over long generations, whereas the PTQ checkpoints often fail to complete them, despite GPTQ achieving lower held-out KL divergence than Standard QAT. Full results are provided in Appendix G.

Comparison with Google's QAT checkpoints. Table 3 compares QUASAR with Google's public QAT checkpoint and PTQ baselines for Gemma-4 E4B. For the INT4 Gemma model, QUASAR achieves a held-out KL divergence of 0.022 and an average accuracy of 35.8 on the ten downstream benchmarks, compared with 0.064 and 34.0 for Google's QAT checkpoint. This corresponds to 66% lower KL divergence and 1.8 points higher average accuracy. Notably, QUASAR achieves these results with only 600M tokens of distillation and 5 hours of training, while Google does not disclose the training data or compute used for its QAT checkpoint. QUASAR also uses a larger quantization group size of 64, compared with 32 for Google's checkpoint. Appendix H extends this comparison to Gemma-4 12B.

5.3 Ablations and Training Overhead

We perform experiments to ablate the effectiveness of the components of QUASAR. Figure 4 presents ablations of QUASAR's two components: scale search and saliency for weighted least

Table 3: Comparison of QUASAR with QAT and PTQ baselines across NVFP4, INT4, and Q4_0 (llama.cpp GGUF format). We report fidelity to BF16 and accuracy on ten reasoning, coding, and long-horizon benchmarks.

				Fidelity		Accuracy (%) ↑										
						Hard reasoning and coding				Long horizon						
Model	Method	Format	bpw	KL ↓	Top-1 ↑	HMMT'26	AIME'25	SuperGPQA	OJBench	MRCR	MRCR ≥64k	MRCR 8-needle	LongProc	LongBench-v2	RULER	**Avg.**
Qwen3-8B	BF16	BF16	16	–	–	44.6	68.4	48.8	22.4	25.8	19.2	16.2	48.9	38.4	80.5	41.3
	RTN	NVFP4 W4A4	4.50	.053	93.0	37.4	**64.6**	45.6	14.7	22.4	15.2	14.1	39.3	36.0	76.6	36.6
	GPTQ	NVFP4 W4A4	4.50	.039	94.0	40.3	64.0	46.2	16.8	23.6	14.6	15.2	39.5	36.0	78.5	37.5
	Standard QAT	NVFP4 W4A4	4.50	.041	94.0	39.0	61.2	45.7	**19.4**	21.6	14.1	12.9	36.5	**37.4**	77.9	36.6
	QUASAR (ours)	NVFP4 W4A4	4.50	**.036**	**94.3**	**40.8**	63.6	**46.8**	18.5	**24.3**	**16.4**	**15.8**	**40.3**	33.8	**78.9**	**37.9**
Qwen3.5-9B	BF16	BF16	16	–	–	47.9	74.5	58.8	19.4	79.4	74.5	53.9	66.8	43.7	90.1	60.9
	RTN	NVFP4 W4A4	4.50	.044	93.2	21.9	37.3	49.1	6.5	54.4	42.6	32.1	35.5	23.1	**91.4**	39.4
	GPTQ	NVFP4 W4A4	4.50	.029	94.5	20.0	29.9	48.6	3.4	56.8	44.6	32.6	42.1	24.3	91.3	39.4
	Standard QAT	NVFP4 W4A4	4.50	.033	94.2	43.1	65.3	55.5	**13.4**	63.7	52.3	39.7	59.0	43.9	89.8	52.6
	QUASAR (ours)	NVFP4 W4A4	4.50	**.028**	**94.6**	**43.8**	**68.8**	**56.0**	**13.4**	**71.7**	**63.8**	**45.0**	**59.7**	**44.1**	90.5	**55.7**
Muse-Glimmer-30B	BF16	BF16	16	–	–	71.7	92.2	60.1	25.0	46.1	38.0	24.9	67.0	60.4	88.4	57.4
	RTN	NVFP4 W4A16	4.50	.054	93.2	69.5	87.7	58.9	25.0	44.3	35.6	26.2	55.7	**59.4**	88.3	55.1
	GPTQ	NVFP4 W4A4	4.50	.066	92.4	66.4	90.3	58.5	25.4	43.1	**40.7**	25.5	60.1	57.3	84.7	55.2
	QUASAR (ours)	NVFP4 W4A16	4.50	**.018**	**96.1**	**72.3**	**90.8**	**59.3**	**27.2**	**48.5**	40.0	**28.5**	**67.3**	58.4	**90.2**	**58.2**
	QUASAR (ours)	NVFP4 W4A4	4.50	.053	93.2	68.3	88.0	58.9	26.3	44.1	35.4	25.5	62.8	54.5	89.7	55.3
Gemma-4 E4B	BF16	BF16	16	–	–	32.8	40.9	42.1	21.6	26.7	15.4	12.6	56.7	42.9	89.7	38.1
	RTN	INT4 g64	4.25	.079	91.5	24.4	31.8	39.0	16.4	22.0	9.3	11.8	35.1	**41.7**	87.5	31.9
	Google QAT	INT4 g32	4.50	.064	92.6	28.6	33.3	39.7	17.7	21.0	8.0	11.5	51.1	39.8	**89.6**	34.0
	QUASAR (ours)	INT4 g64	4.25	**.022**	**95.5**	**29.0**	**37.5**	**40.5**	**18.1**	**24.4**	**12.0**	**12.7**	**53.6**	40.4	89.5	**35.8**
	RTN	GGUF Q4_0	4.50	.194	85.7	18.8	32.8	36.5	14.2	22.9	9.1	11.0	39.0	38.0	83.8	30.6
	Google QAT	GGUF Q4_0	4.50	.088	90.6	**29.0**	35.9	40.8	**19.4**	22.9	9.4	12.8	**57.8**	39.0	89.3	35.6
	imatrix (PTQ)	GGUF Q4_0	4.52	.067	91.1	28.3	**40.1**	**40.9**	19.0	**24.7**	11.7	**13.0**	46.2	39.0	88.9	35.2
	QUASAR (ours)	GGUF Q4_0	4.50	**.044**	**92.8**	27.8	37.6	39.6	**19.4**	24.4	**11.8**	12.7	53.1	**40.4**	**89.6**	**35.6**

squares. Removing both recovers Standard QAT. The experiment uses Qwen3-4B-Thinking-2507 at INT2 with the healing recipe of Section 5.1. Scale search provides most of the gain, while Adam and Fisher saliency perform similarly, despite Adam being computationally cheaper. QUASAR achieves the lowest training and evaluation loss across ablations. Appendix E, Table 8 shows further details.

Furthermore, we compare the efficiency of QUASAR and Standard QAT. QUASAR uses loss-aware reconstruction which introduces slight latency overhead. Table 4 in Appendix presents the average latency breakdown for QAD of Qwen3-4B at INT3 on 8x H100 GPUs. QUASAR only adds 1.4% wall-clock step time compared with Standard QAT (2.97 and 2.93 seconds per step, respectively), with the overhead only coming from the weight reconstruction. Compared with Standard QAT, QUASAR introduces no memory overhead.

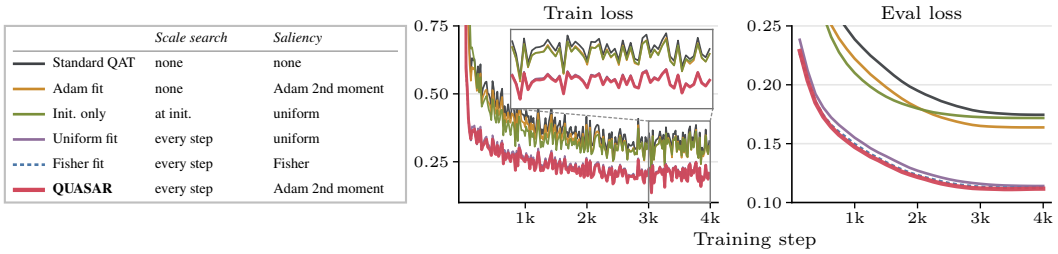


Figure 4: Ablation Study: INT2 QAD for Qwen3-4B-Thinking-2507 with different choices for QUASAR's components. Training and evaluation (held-out) loss both use forward KL.

6 Conclusion

We introduce QUASAR, which integrates loss-aware reconstruction into the training loop to lower the loss floor of QAT. QUASAR combines scale search with a saliency-weighted dequantization fit, using an exponential moving average of squared gradients as the saliency signal. It adds only 1.4% latency overhead per training step and incurs no inference overhead. Our theoretical analysis shows that loss-aware reconstruction error directly controls QAT convergence and the quantized model's final loss, providing a principled foundation for QUASAR's objective. We evaluate QUASAR across four model families and four quantization formats. Across these settings, QUASAR achieves lower training and evaluation loss than competitive QAT baselines and improves downstream accuracy over QAT and PTQ methods. Notably, QUASAR's INT4 Gemma-4 checkpoint outperforms Google's QAT checkpoint in KL divergence and benchmark accuracy.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing and formatting, figure preparation, and literature search. The core idea, methodology, and theoretical development of QUASAR were developed by the authors. All AI-assisted work was reviewed and validated by the authors, who take responsibility for the final content of the work.

REPRODUCIBILITY STATEMENT

Algorithm 1 states the QUASAR reconstruction, and Appendix A.1 gives the variant for each deployment format. Appendix B contains complete proofs of Theorem 1 and Appendix C the experiments that test its assumptions. Appendix D lists all hyperparameters, training budgets, and the evaluation protocol, and Appendices E, F, and G report the full per-task results, and Appendix H lists every public checkpoint we compare against. All models (Qwen3, Qwen3.5, Llama-3.1, Gemma-4, Muse-Glimmer) and datasets (Open-PerfectBlend, OpenMathReasoning, WildChat, and the evaluation benchmarks) are publicly available. Source code for QUASAR, including the training harness and the NVFP4 path, and the quantized checkpoints will be released publicly after the review period.

REFERENCES

Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of SGD with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020.

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 2025. URL https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024. URL https://arxiv.org/abs/2412.15204.

bartowski. Qwen_Qwen3.5-4B-GGUF. https://huggingface.co/bartowski/Qwen_Qwen3.5-4B-GGUF, 2026.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429, 2023.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, 2025a. URL https://aclanthology.org/2025.acl-long.498/.

Mengzhao Chen, Chaoyi Zhang, Jing Liu, Yutao Zeng, Zeyue Xue, Zhiheng Liu, Yunshui Li, Jin Ma, Jie Huang, Xun Zhou, and Ping Luo. Scaling law for quantization-aware training. *arXiv preprint arXiv:2505.14302*, 2025b. URL https://arxiv.org/abs/2505.14302.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

cosmicproc. Qwen3.5-4B-NVFP4. https://huggingface.co/cosmicproc/Qwen3.5-4B-NVFP4, 2026.

cyankiwi. Qwen3.5-4B-AWQ-4bit. https://huggingface.co/cyankiwi/Qwen3.5-4B-AWQ-4bit, 2026.

Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, 2020.

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

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. Accuracy is not all you need. *Advances in Neural Information Processing Systems*, 37:124347–124390, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

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

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*, 2025.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Google. Gemma-4-12B-it. https://huggingface.co/google/gemma-4-12B-it, 2026a.

Google. Gemma-4-E4B-it. https://huggingface.co/google/gemma-4-E4B-it, 2026b.

Google. Gemma-4-12B-it-qat-w4a16-ct. https://huggingface.co/google/gemma-4-12B-it-qat-w4a16-ct, 2026c.

Google. Gemma-4-12B-it-qat-q4_0-gguf. https://huggingface.co/google/gemma-4-12B-it-qat-q4_0-gguf, 2026d.

Google. Gemma-4-E4B-it-qat-w4a16-ct. https://huggingface.co/google/gemma-4-E4B-it-qat-w4a16-ct, 2026e.

Google. Gemma-4-E4B-it-qat-q4_0-gguf. https://huggingface.co/google/gemma-4-E4B-it-qat-q4_0-gguf, 2026f.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pp. 164–171, 1992. URL https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021b.

HMMT. Harvard–MIT mathematics tournament (HMMT). https://www.hmmt.org.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2403.07974.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak–łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Tanishq Kumar, Zachary Ankner, Benjamin Frederick Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.04330.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, pp. 611–626, 2023.

Maxime Labonne. Open-PerfectBlend: An open reproduction of the instruction mixture of "the perfect blend". https://huggingface.co/datasets/mlabonne/open-perfectblend, 2024. Hugging Face dataset; reproduces the mixture of arXiv:2409.20370.

Dongyue Li, Zechun Liu, Kai Yi, Zhenshuo Zhang, Changsheng Zhao, Raghuraman Krishnamoorthi, Harshit Khaitan, Hongyang R. Zhang, and Steven Li. WinQ: Accelerating quantization-aware training of language models around saddle points. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yu Xin Li, Felix Dangel, Derek Tam, and Colin Raffel. Fishers for free? approximating the fisher information matrix by recycling the squared gradient accumulator. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=m3zrHhiCCj.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring llm low-bit quantization degradation for mathematical reasoning. *arXiv preprint arXiv:2501.03035*, 2025b.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

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, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. Quantization hurts reasoning? an empirical study on quantized reasoning models. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=BM192Ps5Nv.

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

Zechun Liu, Changsheng Zhao, 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*, 2025b.

M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines, 2025. URL https://arxiv.org/abs/2502.14739.

Kelly Marchisio, Saurabh Dash, Hongyu Chen, Dennis Aumiller, Ahmet Üstün, Sara Hooker, and Sebastian Ruder. How does quantization affect multilingual llms? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15928–15947, 2024.

Federico Marcuzzi, Xuefei Ning, Roy Schwartz, and Iryna Gurevych. How quantization shapes bias in large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 363–404, 2026.

Mathematical Association of America. American invitational mathematics examination (AIME). https://maa.org/maa-invitational-competitions.

Anmol Mekala, Anirudh Atmakuru, Yixiao Song, Marzena Karpinska, and Mohit Iyyer. Does quantization affect models' performance on long-context tasks? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9422–9470, 2025. URL https://aclanthology.org/2025.emnlp-main.479/.

Meta Superintelligence Lab. Muse-Glimmer-30B. https://huggingface.co/meta-models/Muse-Glimmer-30B, 2026.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7197–7206. PMLR, 2020. URL https://proceedings.mlr.press/v119/nagel20a.html.

NVIDIA. Muse-Glimmer-30B-NVFP4. https://huggingface.co/nvidia/Muse-Glimmer-30B-NVFP4, 2026.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, et al. Pretraining large language models with NVFP4. *arXiv preprint arXiv:2509.25149*, 2025.

OpenAI. OpenAI MRCR: Long context multiple needle in a haystack benchmark. Hugging Face dataset https://huggingface.co/datasets/openai/mrcr, 2025. Extends the MRCR evaluation of Vodrahalli et al. (2024).

Boris T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

Qwen Team. Qwen3.5-4B. https://huggingface.co/Qwen/Qwen3.5-4B, 2026a.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026b. URL https://qwen.ai/blog?id=qwen3.5.

Red Hat AI. Muse-Glimmer-30B-NVFP4. https://huggingface.co/RedHatAI/Muse-Glimmer-30B-NVFP4, 2026a.

Red Hat AI. Qwen3.5-4B-quantized.w4a16. https://huggingface.co/RedHatAI/Qwen3.5-4B-quantized.w4a16, 2026b.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43445–43460. PMLR, 2024. URL https://proceedings.mlr.press/v235/sardana24a.html.

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*, 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*, 2024b.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Unsloth. Gemma-4-12b-it-GGUF. https://huggingface.co/unsloth/gemma-4-12b-it-GGUF, 2026a.

Unsloth. Gemma-4-12B-it-qat-GGUF. https://huggingface.co/unsloth/gemma-4-12B-it-qat-GGUF, 2026b.

Unsloth. Gemma-4-E4B-it-GGUF. https://huggingface.co/unsloth/gemma-4-E4B-it-GGUF, 2026c.

Unsloth. Gemma-4-E4B-it-qat-GGUF. https://huggingface.co/unsloth/gemma-4-E4B-it-qat-GGUF, 2026d.

vLLM project. LLM Compressor. https://github.com/vllm-project/llm-compressor, 2024.

vLLM Project. GPTQModel. https://docs.vllm.ai/en/stable/features/quantization/gptqmodel/, 2026a. Accessed September 24, 2026.

vLLM Project. NVIDIA Model Optimizer. https://docs.vllm.ai/en/stable/features/quantization/modelopt/, 2026b. Accessed September 24, 2026.

vLLM Project. compressed_tensors_w4a16_mxfp4. https://docs.vllm.ai/en/latest/api/vllm/model_executor/layers/quantization/compressed_tensors/schemes/compressed_tensors_w4a16_mxfp4/, 2026c. Accessed September 24, 2026.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, Orhan Firat, Angeliki Lazaridou, Jean-Baptiste Lespiau, Nithya Attaluri, and Kate Olszewska. Michelangelo: Long context evaluations beyond haystacks via latent structure queries, 2024.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Zhexu Wang, Yiping Liu, Yejie Wang, Wenyang He, Bofei Gao, Muxi Diao, Yanxu Chen, Kelin Fu, Flood Sung, Zhilin Yang, Tianyu Liu, and Weiran Xu. OJBench: A competition level code benchmark for large language models, 2025. URL https://arxiv.org/abs/2506.16395.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. PyHessian: Neural networks through the lens of the hessian. In *IEEE International Conference on Big Data*, 2020.

Chengxi Ye, Grace Chu, Yanfeng Liu, Yichi Zhang, Lukasz Lew, Li Zhang, Mark Sandler, and Andrew G. Howard. Robust training of neural networks at arbitrary precision and sparsity. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10008354.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. LongProc: Benchmarking long-context language models on long procedural generation. In *Conference on Language Modeling (COLM)*, 2025.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations*, 2019.

YoozLabs. Qwen3.5-4B-qat-GGUF. https://huggingface.co/YoozLabs/Qwen3.5-4B-qat-GGUF, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

APPENDIX

A METHOD DETAILS

This appendix provides additional details on QUASAR. Using real weight groups, it examines per-weight reconstruction error, the clipping ranges selected by scale search, and the saliency signal h, complementing Section 3. It also reports training-step overhead (Table 4), gives algorithm variants for formats with fast inference kernels (Appendix A.1), and compares two saliency weightings (Appendix A.2).

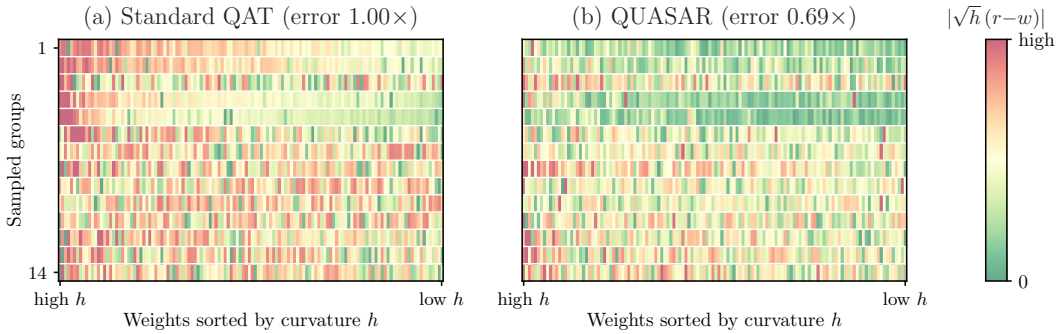
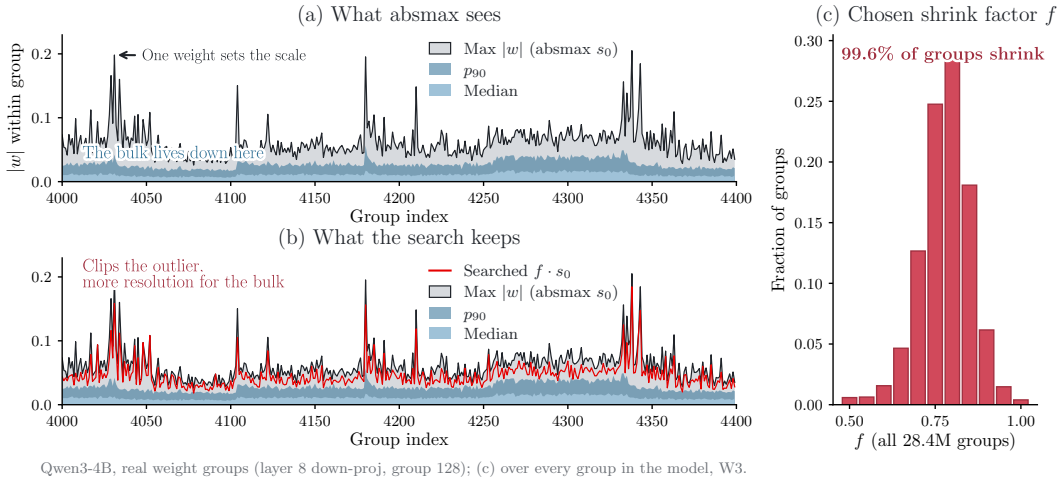


Figure 5: Per-weight loss-aware reconstruction error for 14 Qwen3-4B weight groups at INT3. Color shows $|\sqrt{h}\,(r-w)|$ under (a) Standard QAT with min–max reconstruction and (b) QUASAR. Each row contains 128 weights ordered by saliency h.

At INT3, 99.6% of Qwen3-4B weight groups select a range narrower than min–max, reducing $\widehat{S}_t$ to 69% of the full-range baseline (Figure 6). Figure 7 shows the selected ranges across models and bit widths.



Qwen3-4B, real weight groups (layer 8 down-proj, group 128); (c) over every group in the model, W3.

Figure 6: Scale search for Qwen3-4B weight groups at INT3. (a) One extreme weight sets the min–max range. (b) QUASAR clips the outlier and covers the bulk. (c) Distribution of selected range factors $f^\star$ across 28.4M groups; 99.6% use a range narrower than min–max.

Across both models, the median selected range factor decreases from 0.95 at INT4 to 0.60 at INT2 (Figure 7). With fewer codes, the search clips extreme weights more aggressively to better represent the bulk of the distribution.

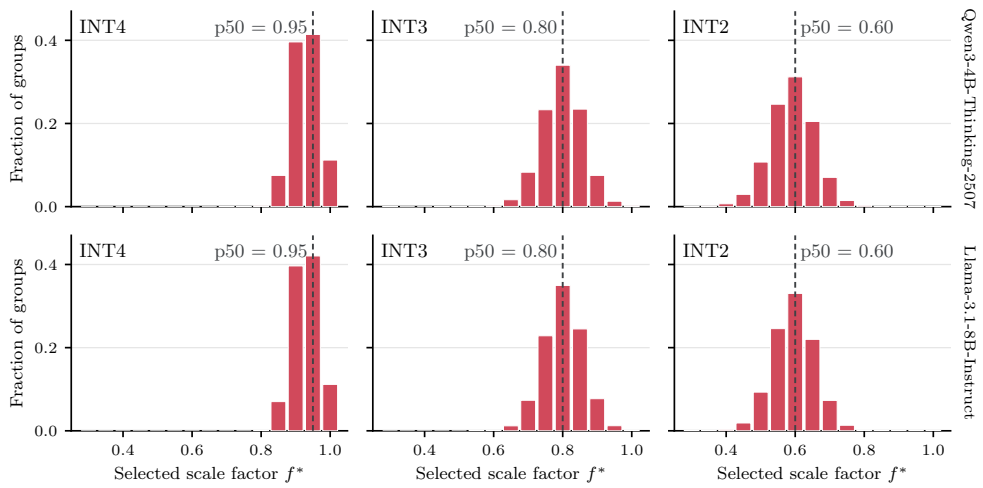


Figure 7: Distribution of selected clipping-range factors $f^\star$ across weight groups after QAD at INT4, INT3, and INT2 for Qwen3-4B-Thinking-2507 and Llama-3.1-8B-Instruct. Lower bit widths favor narrower ranges.

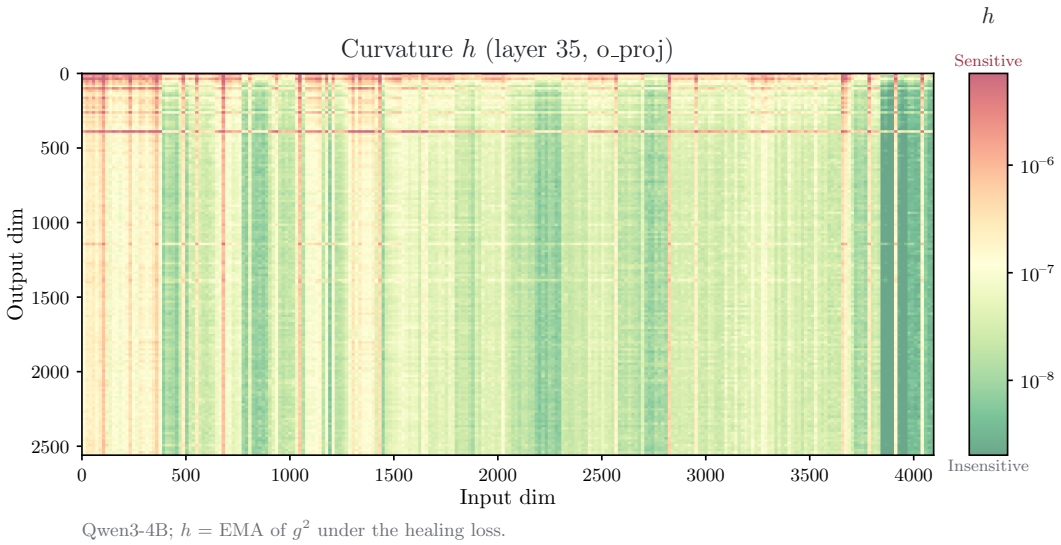


Figure 8: Saliency h in one Qwen3-4B layer. Weight groups span 128 weights along the input dimension. The large variation within each group lets QUASAR prioritize the most salient weights.

Table 4: Wall-clock time of one training step, per component and in seconds: quantization-aware distillation of Qwen3-4B-Thinking-2507 at INT3 on 8x H100 GPUs. QUASAR differs from Standard QAT only in the weight reconstruction row.

Component	Standard QAT	QUASAR
Teacher forward	0.373	0.373
Student compute	0.366	0.366
Weight reconstruction	**0.294**	**0.335**
Loss computation	0.065	0.065
Backward	1.805	1.805
Optimizer update	0.021	0.022
Other + synchronization	0.006	0.006
Total step	**2.930**	**2.972**
% vs. Standard QAT	–	+1.4%

A.1 Algorithm Statements for Deployment Formats

QUASAR applies to affine and symmetric integer formats, NVFP4, and MXFP4 (Rouhani et al., 2023). Each variant follows Algorithm 1: it searches over code assignments, fits the allowed dequantization parameters when the format permits, and selects the reconstruction with the lowest loss-aware error. The code grid and stored parameters depend on the format. Table 5 summarizes the variants.

QUASAR checkpoints use standard layouts supported by vLLM's quantization backends. In vLLM, Marlin (Frantar et al., 2025) and Machete (vLLM Project, 2026a) serve INT4 weights, NVFP4 runs natively on Blackwell GPUs (vLLM Project, 2026b), and MXFP4 weight-only checkpoints are supported (vLLM Project, 2026c). INT3 and INT2 currently lack dedicated fast kernels, so we treat them as accuracy frontiers rather than deployment formats.

Table 5: QUASAR variants for deployment formats with fast inference kernels. The formats differ in their code grids, group sizes, stored parameters, and dequantization constraints.

Format	Codes	Group	Stored parameters	Fit	Statement
INT sym. (GPTQ layout)	INT4	32–128	scale	Eq. (7)	Alg. 3
INT affine (AWQ layout)	INT4	32–128	scale, int. zero-point	Eq. (8)	Alg. 4
NVFP4	E2M1	16	E4M3 scale, FP32 tensor scale	Eq. (7), E4M3-rounded	Alg. 5
MXFP4	E2M1	32	E8M0 scale	search only	Alg. 6

Affine integer formats. For affine quantization, $r_i = sq_i + z$. Given codes q, minimizing Equation (3) over s and z is weighted least squares:

$$s^\star = \frac{\sum_i h_i(q_i - \bar{q}_h)(w_i - \bar{w}_h)}{\sum_i h_i(q_i - \bar{q}_h)^2}, \qquad z^\star = \bar{w}_h - s^\star \bar{q}_h, \tag{6}$$

where $\bar{q}_h = \sum_i h_i q_i / \sum_i h_i$ and $\bar{w}_h = \sum_i h_i w_i / \sum_i h_i$ are the saliency-weighted means. Algorithm 2 applies this fit to every candidate code assignment.

Algorithm 2 QUASAR for affine integer quantization. Candidate ranges assign codes; weighted least squares then fits the stored scale and offset.

Require: latent weights $w \in \mathbb{R}^g$ of one group; saliency $h \in \mathbb{R}^g_+$ (Adam's second moment); bit width b; clipping factors $\mathcal{F} \subseteq (0, 1]$

$c \leftarrow \frac{1}{2}(\min_i w_i + \max_i w_i)$; $\rho \leftarrow \frac{1}{2}(\max_i w_i - \min_i w_i)$ ▷ Standard QAT uses $f{=}1$
$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $\tilde{s}_f \leftarrow 2f\rho/(2^b - 1)$; $\tilde{z}_f \leftarrow c - f\rho$ ▷ assignment parameters
  $q_f \leftarrow \mathrm{clamp}_{[0,2^b-1]}(\mathrm{round}((w - \tilde{z}_f \mathbf{1})/\tilde{s}_f))$ ▷ code assignment $w \mapsto q$
  $(s_f, z_f) \leftarrow$ Eq. (6) with $q = q_f$ ▷ dequantization fit
  $r_f \leftarrow s_f q_f + z_f \mathbf{1}$
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f$; $(q, s, z) \leftarrow (q_f, s_f, z_f)$ **end if**
end for
return $r \leftarrow sq + z\mathbf{1}$

All candidates are evaluated in one batched kernel. The assignment parameters are temporary; the best codes and fitted scale and offset define the reconstruction. Backpropagation uses the same STE as standard QAT. This adds 1.4% to the step time (Table 4) and preserves the standard RTN/GPTQ inference layout.

Symmetric integer formats. Symmetric integer formats store one scale per group and no zero-point, so $r_i = sq_i$. For fixed signed codes, weighted least squares gives

$$s^\star = \frac{\sum_i h_i q_i w_i}{\sum_i h_i q_i^2}. \tag{7}$$

This is the GPTQ layout used by W4A16 and W4A8 serving kernels.

Algorithm 3 QUASAR for symmetric integer quantization. Signed codes use a zero-centered assignment, and weighted least squares fits one scale.

Require: latent weights $w \in \mathbb{R}^g$ of one group; saliency $h \in \mathbb{R}^g_+$ (Adam's second moment); bit width b; clipping factors $\mathcal{F} \subseteq (0, 1]$

$M \leftarrow 2^{b-1}$; $\rho \leftarrow \max_i |w_i|$ ▷ Standard QAT uses $f{=}1$
$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $\tilde{s}_f \leftarrow f\rho/M$ ▷ assignment scale
  $q_f \leftarrow \mathrm{clamp}_{[-M, M-1]}(\mathrm{round}(w/\tilde{s}_f))$ ▷ code assignment $w \mapsto q$
  $s_f \leftarrow \sum_i h_i q_{f,i} w_i \,/\, \sum_i h_i q_{f,i}^2$ ▷ Eq. (7); dequantization fit
  $r_f \leftarrow s_f q_f$
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f$; $(q, s) \leftarrow (q_f, s_f)$ **end if**
end for
return $r \leftarrow sq$

Affine integer formats with an integer zero-point. Some affine layouts, including AWQ, store an integer zero-point q_0 and reconstruct weights as $r_i = s(q_i - q_0)$. For each code assignment, QUASAR first computes the continuous fit, rounds $q_0 = -z^\star/s^\star$ to a valid code, and refits the scale:

$$q_0 = \mathrm{clamp}\big(\mathrm{round}(-z^\star/s^\star)\,,\, 0,\, 2^b - 1\big)\,, \qquad s^\star_{q_0} = \frac{\sum_i h_i (q_i - q_0)\, w_i}{\sum_i h_i (q_i - q_0)^2}. \tag{8}$$

Algorithm 4 applies this constraint to every candidate.

Algorithm 4 QUASAR for affine integer quantization with an integer zero-point. The continuous fit is snapped to an integer zero-point, then the scale is refit.

Require: latent weights $w \in \mathbb{R}^g$ of one group; saliency $h \in \mathbb{R}^g_+$ (Adam's second moment); bit width b; clipping factors $\mathcal{F} \subseteq (0, 1]$

$c \leftarrow \frac{1}{2}(\min_i w_i + \max_i w_i)$; $\rho \leftarrow \frac{1}{2}(\max_i w_i - \min_i w_i)$ ▷ Standard QAT uses $f{=}1$
$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $\tilde{s}_f \leftarrow 2f\rho/(2^b - 1)$; $\tilde{z}_f \leftarrow c - f\rho$ ▷ assignment parameters
  $q_f \leftarrow \mathrm{clamp}_{[0, 2^b-1]}(\mathrm{round}((w - \tilde{z}_f \mathbf{1})/\tilde{s}_f))$ ▷ code assignment $w \mapsto q$
  $(\bar{s}_f, \bar{z}_f) \leftarrow$ Eq. (6) with $q = q_f$ ▷ unconstrained fit
  $q_{0,f} \leftarrow \mathrm{clamp}_{[0, 2^b-1]}(\mathrm{round}(-\bar{z}_f/\bar{s}_f))$ ▷ Eq. (8)
  $s_f \leftarrow \sum_i h_i (q_{f,i} - q_{0,f})\, w_i \,/\, \sum_i h_i (q_{f,i} - q_{0,f})^2$ ▷ Eq. (8); dequantization fit
  $r_f \leftarrow s_f (q_f - q_{0,f}\mathbf{1})$
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f$; $(q, s, q_0) \leftarrow (q_f, s_f, q_{0,f})$ **end if**
end for
return $r \leftarrow s(q - q_0\mathbf{1})$

NVFP4. NVFP4 represents groups of 16 weights with E2M1 codes, an E4M3 scale per group, and a tensor-level FP32 scale σ (Alvarez et al., 2025). It is symmetric and has no zero-point. We set σ from the tensor's maximum absolute weight and keep it fixed. For each candidate, QUASAR rounds the assignment scale to E4M3, assigns E2M1 codes, and fits the stored E4M3 dequantization scale. It keeps the reconstruction with the lowest loss-aware error (Algorithm 5).

Algorithm 5 QUASAR for NVFP4. E2M1 codes use E4M3-rounded assignment and fitted group scales under a fixed FP32 tensor scale.

Require: group weights $w \in \mathbb{R}^{16}$; saliency $h \in \mathbb{R}^{16}_{+}$ (Adam's second moment); clipping factors $\mathcal{F} \subseteq (0, 1]$; tensor scale σ (fixed)

$\tilde{w} \leftarrow w/\sigma$; $\rho \leftarrow \max_i |\tilde{w}_i|/6$ ▷ 6 = largest E2M1 magnitude
$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $\tilde{s}_f \leftarrow Q_{\mathrm{E4M3}}(f\rho)$ ▷ assignment scale
  $q_f \leftarrow Q_{\mathrm{E2M1}}(\tilde{w}/\tilde{s}_f)$ ▷ code assignment $w \mapsto q$
  $s_f \leftarrow Q_{\mathrm{E4M3}}\big(\sum_i h_i q_{f,i}\tilde{w}_i \,/\, \sum_i h_i q_{f,i}^2\big)$ ▷ Eq. (7); dequantization fit
  $r_f \leftarrow \sigma s_f q_f$
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f$; $(q, s) \leftarrow (q_f, s_f)$ **end if**
end for
return $r \leftarrow \sigma s q$

MXFP4. MXFP4 stores E2M1 codes in groups of 32 with one E8M0 power-of-two scale per group. Because the stored scale is discrete, QUASAR searches the E8M0 scale grid directly rather than fitting a continuous scale. Saliency h determines which candidate has the lowest loss-aware error (Algorithm 6).

Algorithm 6 QUASAR for MXFP4. E2M1 codes share an E8M0 power-of-two scale, so scale search also determines the stored dequantization scale.

Require: group weights $w \in \mathbb{R}^{32}$; saliency $h \in \mathbb{R}^{32}_{+}$ (Adam's second moment); search depth k

$e_0 \leftarrow$ smallest integer with $2^{e_0} \cdot 6 \geq \max_i |w_i|$ ▷ 6 = largest E2M1 magnitude
$\mathcal{F} \leftarrow \{2^{-\delta} : \delta = 0, 1, \ldots, k\}$
$\widehat{S}^\star \leftarrow \infty$
for $f \in \mathcal{F}$ **do**
  $s_f \leftarrow f 2^{e_0}$ ▷ assignment and dequantization scale
  $q_f \leftarrow Q_{\mathrm{E2M1}}(w/s_f)$ ▷ code assignment $w \mapsto q$
  $r_f \leftarrow s_f q_f$
  $\widehat{S}_f \leftarrow \sum_i h_i (r_{f,i} - w_i)^2$ ▷ loss-aware reconstruction error
  if $\widehat{S}_f < \widehat{S}^\star$ **then** $\widehat{S}^\star \leftarrow \widehat{S}_f$; $(q, s) \leftarrow (q_f, s_f)$ **end if**
end for
return $r \leftarrow sq$

A.2 Choice of saliency weighting

QUASAR weights squared reconstruction error by saliency h_t, our diagonal curvature estimate. A natural alternative is h_t^2, motivated by the squared linearized gradient mismatch:

$$\|H_t(r_t - w_t)\|^2 = (r_t - w_t)^\top H_t^2 (r_t - w_t).$$

We compare both choices with all other settings fixed: AdamW at INT2, INT3, and INT4, and SGD at three stable INT2 learning rates. Weighting by h_t gives lower held-out loss in all six settings. Weighting by h_t^2 is unstable at INT4 and at the higher SGD learning rates (Figure 9).

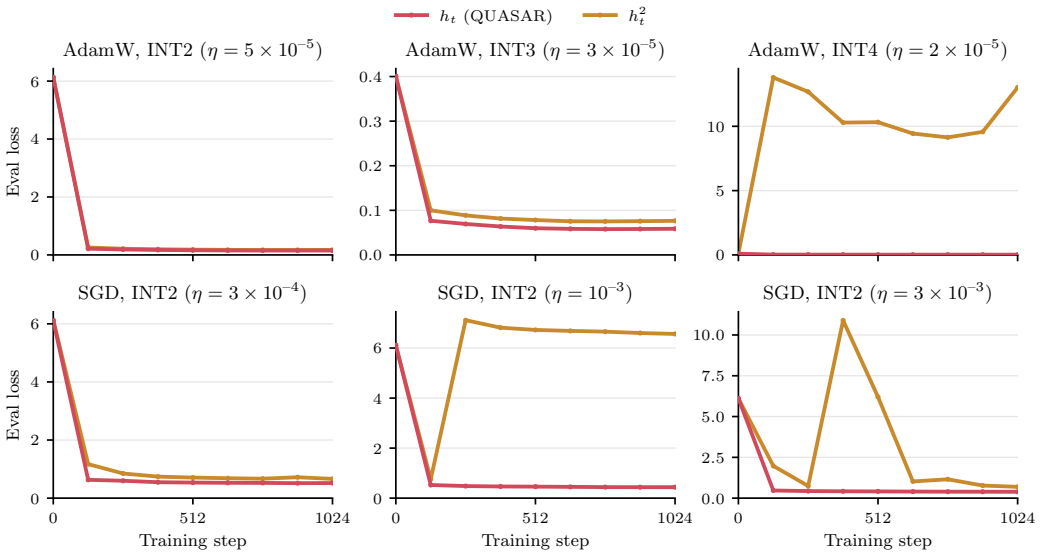


Figure 9: Comparison of saliency weights h_t and h_t^2. We report held-out loss under AdamW at INT2, INT3, and INT4 (top) and under SGD at three INT2 learning rates (bottom).

B Proofs and Deferred Theory Discussion

This appendix contains the proofs of the results stated in Section 4.

B.1 Proofs of the main results

We use the notation $\mathbb{E}_t[\cdot]$ introduced in Section 4 and write $H(y) := \nabla^2\mathcal{L}(y)$ for the Hessian at y. We first record a sufficient condition for Assumption 3. Fix any reconstruction index t under consideration, including the terminal index $t = T$, and put

$$d_t = r_t - w_t, \qquad D_t = \operatorname{diag}(h_t).$$

Suppose that, at every point $y = w_t + ud_t$ on the reconstruction segment,

$$H(y)^2 \preceq CD_t, \qquad 0 \le u \le 1. \tag{9}$$

By the fundamental theorem of calculus, Jensen's inequality, and the symmetry of the Hessian,

$$\begin{aligned}\|\nabla\mathcal{L}(r_t) - \nabla\mathcal{L}(w_t)\|^2 &= \left\|\int_0^1 H(w_t + ud_t)d_t\,du\right\|^2 \\ &\le \int_0^1 d_t^\top H(w_t + ud_t)^2 d_t\,du \\ &\le Cd_t^\top D_t d_t = C\widehat{S}_t.\end{aligned}$$

Thus (9) implies Assumption 3. In particular, if $\mathcal{L}$ is L-smooth and $h_{t,i} \ge h_{\min} > 0$, then, for every $y, d \in \mathbb{R}^d$,

$$\|H(y)d\|^2 \le L^2\|d\|^2 \le \frac{L^2}{h_{\min}}\sum_i h_{t,i}d_i^2.$$

Consequently, Assumption 3 holds with $C = L^2/h_{\min}$.

Proof of Theorem 1. For part (i), set

$$a_t = \nabla\mathcal{L}(w_t), \qquad c_t = \nabla\mathcal{L}(r_t).$$

By smoothness and the update in Assumption 2,

$$\begin{aligned}\mathcal{L}(w_{t+1}) \le \mathcal{L}(w_t) &- \eta\langle a_t, c_t + \xi_t\rangle \\ &+ \frac{L\eta^2}{2}\|c_t + \xi_t\|^2.\end{aligned}$$

Conditional on $\mathcal{H}_t$, both a_t and c_t are fixed. Moreover,

$$\mathbb{E}_t\|c_t + \xi_t\|^2 = \|c_t\|^2 + \mathbb{E}_t\|\xi_t\|^2 \le (1+M)\|c_t\|^2 + \sigma^2.$$

It follows that

$$\mathbb{E}_t[\mathcal{L}(w_{t+1})] \le \mathcal{L}(w_t) - \eta\langle a_t, c_t\rangle + \frac{L\eta^2}{2}\left((1+M)\|c_t\|^2 + \sigma^2\right).$$

Using the identity

$$2\langle a_t, c_t\rangle = \|a_t\|^2 + \|c_t\|^2 - \|c_t - a_t\|^2$$

and Assumption 3, we obtain

$$\begin{aligned}\mathbb{E}_t[\mathcal{L}(w_{t+1})] \le \mathcal{L}(w_t) &- \frac{\eta}{2}\|a_t\|^2 \\ &- \frac{\eta\big(1 - L\eta(1+M)\big)}{2}\|c_t\|^2 + \frac{\eta C}{2}\widehat{S}_t + \frac{L\eta^2}{2}\sigma^2.\end{aligned} \tag{10}$$

We take expectations and sum this inequality for $0 \le t < T$. The terms involving the loss telescope, and therefore

$$\begin{aligned}\frac{\eta}{2}\sum_{t=0}^{T-1}\mathbb{E}\Big[\|\nabla\mathcal{L}(w_t)\|^2 + \big(1 - L\eta(1+M)\big)\|\nabla\mathcal{L}(r_t)\|^2\Big] &\le \mathcal{L}(w_0) - \mathbb{E}\mathcal{L}(w_T) \\ &+ \frac{\eta C}{2}\sum_{t=0}^{T-1}\mathbb{E}\widehat{S}_t + \frac{L\eta^2 T}{2}\sigma^2.\end{aligned}$$

Since $\mathcal{L}(w_T) \geq \mathcal{L}^\star$, the desired result follows after division by $\eta T/2$. This proves part (i).

For part (ii), put $\gamma = 1-\eta\mu$. The assumption on η ensures that $0 < \gamma < 1$ and that $1-L\eta(1+M) > 0$. We may thus omit the term involving $\|c_t\|^2$ in (10). The PL inequality at w_t gives

$$\mathbb{E}[\mathcal{L}(w_{t+1}) - \mathcal{L}^\star] \leq \gamma\,\mathbb{E}[\mathcal{L}(w_t) - \mathcal{L}^\star] + \frac{\eta C}{2}\mathbb{E}\widehat{S}_t + \frac{L\eta^2}{2}\sigma^2.$$

Iteration yields

$$\begin{aligned}\mathbb{E}[\mathcal{L}(w_T) - \mathcal{L}^\star] \leq \gamma^T(\mathcal{L}(w_0) - \mathcal{L}^\star) \\ + \frac{\eta C}{2}\sum_{t=0}^{T-1}\gamma^{T-1-t}\mathbb{E}\widehat{S}_t \\ + \frac{L\eta\sigma^2}{2\mu}\big(1-\gamma^T\big).\end{aligned} \tag{11}$$

It remains to compare w_T and r_T. Applying the smoothness inequality at w_T with the point $w_T - L^{-1}\nabla\mathcal{L}(w_T)$, we get

$$\mathcal{L}\left(w_T - \frac{1}{L}\nabla\mathcal{L}(w_T)\right) \leq \mathcal{L}(w_T) - \frac{1}{2L}\|\nabla\mathcal{L}(w_T)\|^2.$$

The left-hand side is at least $\mathcal{L}^\star$, and hence

$$\|\nabla\mathcal{L}(w_T)\|^2 \leq 2L\big(\mathcal{L}(w_T) - \mathcal{L}^\star\big).$$

Assumption 3 at the terminal index $t = T$ gives

$$\|\nabla\mathcal{L}(r_T) - \nabla\mathcal{L}(w_T)\|^2 \leq C\widehat{S}_T.$$

By the PL inequality at r_T and $\|x+y\|^2 \leq 2\|x\|^2 + 2\|y\|^2$, it follows that

$$\begin{aligned}\mathcal{L}(r_T) - \mathcal{L}^\star &\leq \frac{1}{2\mu}\|\nabla\mathcal{L}(r_T)\|^2 \\ &\leq \frac{1}{\mu}\|\nabla\mathcal{L}(w_T)\|^2 + \frac{1}{\mu}\|\nabla\mathcal{L}(r_T) - \nabla\mathcal{L}(w_T)\|^2 \\ &\leq \frac{2L}{\mu}\big(\mathcal{L}(w_T) - \mathcal{L}^\star\big) + \frac{C}{\mu}\widehat{S}_T.\end{aligned}$$

Taking expectations and substituting (11) yields the exact intermediate bound

$$\begin{aligned}\mathbb{E}[\mathcal{L}(r_T) - \mathcal{L}^\star] \leq \frac{2L}{\mu}\gamma^T(\mathcal{L}(w_0) - \mathcal{L}^\star) \\ + \frac{L\eta C}{\mu}\sum_{t=0}^{T-1}\gamma^{T-1-t}\mathbb{E}\widehat{S}_t \\ + \frac{L^2\eta\sigma^2}{\mu^2}(1-\gamma^T) + \frac{C}{\mu}\mathbb{E}\widehat{S}_T.\end{aligned} \tag{12}$$

The random index τ in part (ii) is independent of the training randomness and has distribution

$$\Pr(\tau = t) = \frac{\eta\mu\gamma^{T-1-t}}{1-\gamma^T}, \qquad t \in \{0, \ldots, T-1\},$$

so later training steps receive greater weight. Because $1-\gamma = \eta\mu$, these probabilities sum to one, and independence gives

$$\mathbb{E}\widehat{S}_\tau = \sum_{t=0}^{T-1}\frac{\eta\mu\gamma^{T-1-t}}{1-\gamma^T}\mathbb{E}\widehat{S}_t,$$

$$\sum_{t=0}^{T-1}\gamma^{T-1-t}\mathbb{E}\widehat{S}_t = \frac{1-\gamma^T}{\eta\mu}\mathbb{E}\widehat{S}_\tau.$$

Thus the second term in (12) equals $\frac{LC}{\mu^2}(1-\gamma^T)\mathbb{E}\widehat{S}_\tau$. Since the reconstruction errors are nonnegative and $1-\gamma^T \leq 1$, this is at most $\frac{LC}{\mu^2}\mathbb{E}\widehat{S}_\tau$. Applying the same bound to the noise term and recalling $\gamma = 1-\eta\mu$ gives (5), proving part (ii). □

B.2 Deferred discussion of the assumptions and the bound

Remark 1 (Assumptions). Assumptions 1 and 2 are standard in stochastic optimization. Assumption 1 requires a twice continuously differentiable, smooth objective with a finite infimum, but not convexity. Assumption 2 allows gradient noise to grow with the gradient norm and recovers bounded variance when $M = 0$; the same condition is used to analyze biased SGD (Ajalloeian & Stich, 2020).

Assumption 3 is specific to reconstruction: small loss-aware reconstruction error must imply a small gradient mismatch. Appendix B.1 derives a sufficient curvature condition. Under L-smoothness and a positive saliency floor $h_{t,i} \geq h_{\min} > 0$, it holds with $C = L^2/h_{\min}$ (Ghadimi & Lan, 2013; Bottou et al., 2018; Ajalloeian & Stich, 2020). Adam's second moment motivates the saliency estimate but does not guarantee the condition. Empirically, saliency tracks the Hessian diagonal, and the full quadratic form is nearly proportional to $\widehat{S}$ (Figure 15, Appendix C).

Assumption 4 is the Polyak–Łojasiewicz (PL) condition and is needed only for part (ii) of Theorem 1. Strong convexity implies PL, but PL does not require convexity and ensures that every stationary point is a global minimum (Polyak, 1963; Karimi et al., 2016). Local PL-type conditions have also been established for some overparameterized neural networks (Liu et al., 2022).

Comparison with standard and biased SGD. To compare the guarantees, define

$$B_T := \frac{\mathcal{L}(w_0) - \mathcal{L}^\star}{\eta T} + \eta\sigma^2, \qquad \overline{S}_T := \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\widehat{S}_t.$$

Let $\alpha_\eta := 1 - L\eta(1+M) > 0$. Up to constant factors, the relevant bounds are

$$\begin{aligned}
\text{Standard SGD:} \quad & \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla\mathcal{L}(w_t)\|^2 \lesssim B_T, \\
\text{Biased SGD:} \quad & \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla\mathcal{L}(w_t)\|^2 \lesssim \frac{B_T + \zeta^2}{1-m}, \\
\text{QUASAR:} \quad & \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\nabla\mathcal{L}(w_t)\|^2 + \alpha_\eta\|\nabla\mathcal{L}(r_t)\|^2\big] \lesssim B_T + C\overline{S}_T.
\end{aligned} \tag{13}$$

Biased-SGD analysis assumes $\|b_t\|^2 \leq m\|\nabla\mathcal{L}(w_t)\|^2 + \zeta^2$ with $0 \leq m < 1$ (Ajalloeian & Stich, 2020). In QAT, the bias is the reconstruction-induced gradient mismatch $b_t = e_t$. Assumption 3 bounds this mismatch by $C\widehat{S}_t$, which QUASAR minimizes at every step. The resulting bound contains the usual SGD terms plus $C\overline{S}_T$ and controls stationarity at both the latent and reconstructed weights.

C EMPIRICAL VALIDATION OF THE THEORY

This appendix tests the assumptions and predictions of Section 4. Across both model families and all tested bit widths, end-of-training reconstruction error closely tracks held-out KL for Standard QAT, Denoising QAT, and QUASAR. QUASAR attains the lowest value of both in every setting (Figure 10; per-projection results in Figure 11). The same ordering holds under the plain-SGD dynamics analyzed by the theorem (Figure 18). The relationship is also causal: restricting the scale search raises both $\widehat{S}_T$ and final KL. It is predictive before training as well: reconstruction error at initialization predicts final KL (Figures 12 and 13).

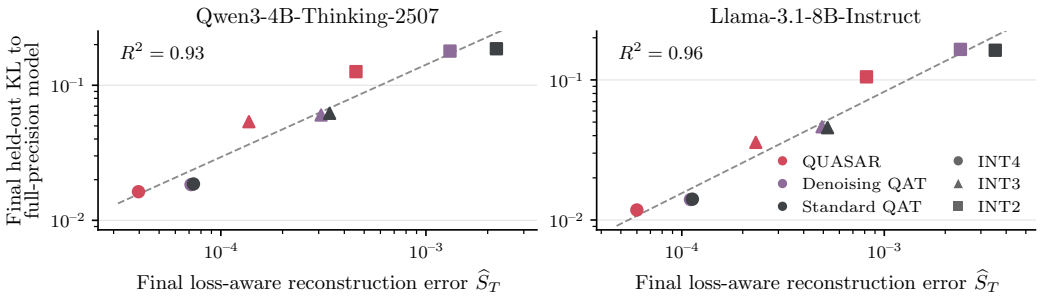


Figure 10: **Reconstruction error tracks final KL.** End-of-training loss-aware reconstruction error versus final held-out KL to the full-precision teacher, for Standard QAT, Denoising QAT, and QUASAR at INT4, INT3, and INT2.

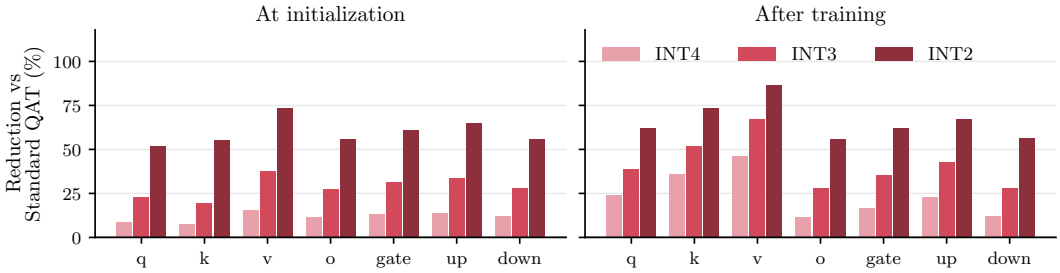


Figure 11: **QUASAR reduces reconstruction error across projection types.** Median reduction in loss-aware reconstruction error relative to Standard QAT across Qwen3-4B-Thinking-2507 modules at INT4, INT3, and INT2, measured at initialization (left) and after training (right). Error is measured against the full-precision weights.

Validation under SGD dynamics. Theorem 1 analyzes STE–SGD, whereas the main experiments use AdamW. We therefore repeat the INT2 quantization-aware distillation experiment with plain SGD for 1,024 steps at global batch 32, using a constant learning rate and no momentum, warmup, or gradient clipping. QUASAR replaces AdamW's second moment with an explicit EMA of squared gradients; the remaining recipe is unchanged. At every stable learning rate, QUASAR reaches lower training and held-out loss floors than Standard QAT, preserving the ordering observed with AdamW (Figure 18).

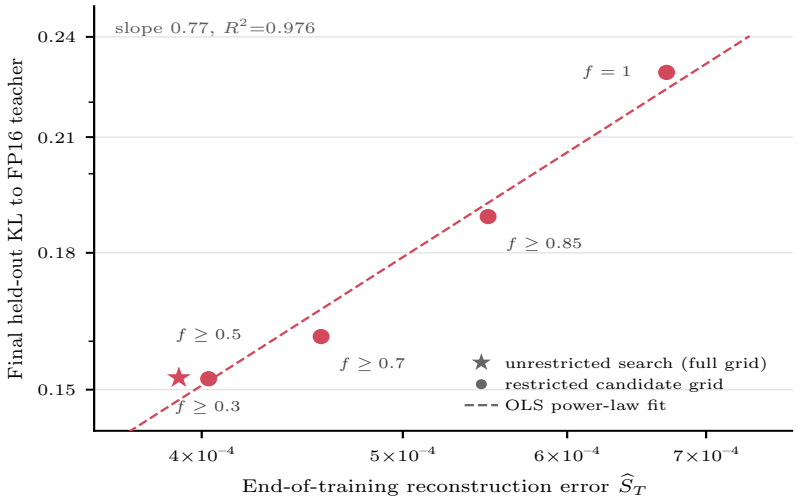


Figure 12: **Reconstruction error drives final loss.** Five QUASAR INT2 runs differ only in the lower bound imposed on the scale-search factor f. Restricting the search raises both the final reconstruction error $\widehat{S}_T$ and held-out KL after 1,000 training steps. The star marks the unrestricted search.

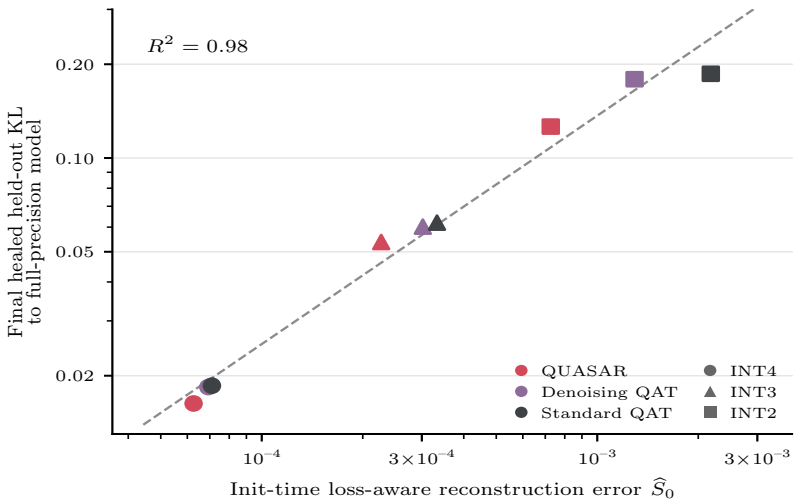


Figure 13: **Reconstruction error predicts final loss before training.** Across Standard QAT, Denoising QAT, and QUASAR at INT4, INT3, and INT2, reconstruction error at initialization predicts held-out KL to the full-precision model after 4,096 steps ($R^2 = 0.98$). Results use Qwen3-4B-Thinking-2507.

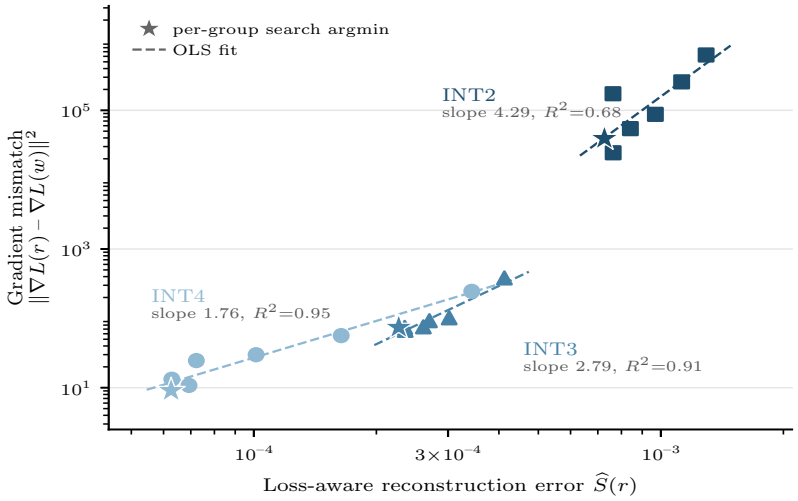


Figure 14: **Reconstruction error controls gradient mismatch.** Across INT4, INT3, and INT2, the squared gradient mismatch rises with the loss-aware reconstruction error $\widehat{S}$, supporting Assumption 3. Circles, triangles, and squares show fixed clipping factors for INT4, INT3, and INT2, respectively; stars show QUASAR's per-group scale-search solutions.

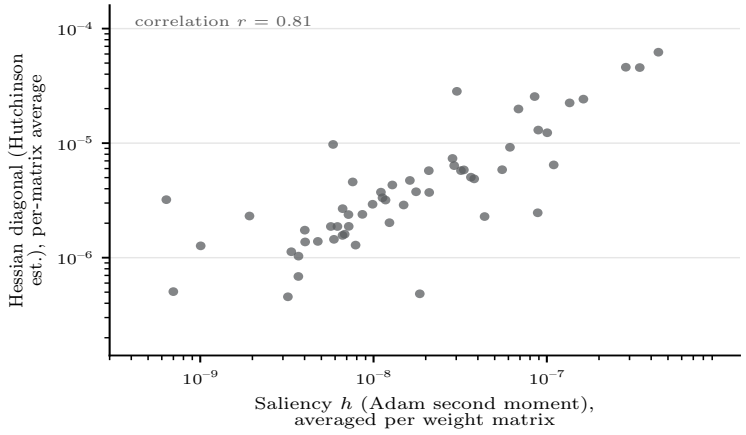


Figure 15: **QUASAR's saliency tracks curvature.** Each point represents one weight matrix from a trained INT2 Qwen3-4B-Thinking checkpoint and compares its mean saliency h (Adam's second moment) with a 32-sample Hutchinson estimate of its mean Hessian diagonal. The log-scale correlation is $r = 0.81$ (Dong et al., 2020; Yao et al., 2020).

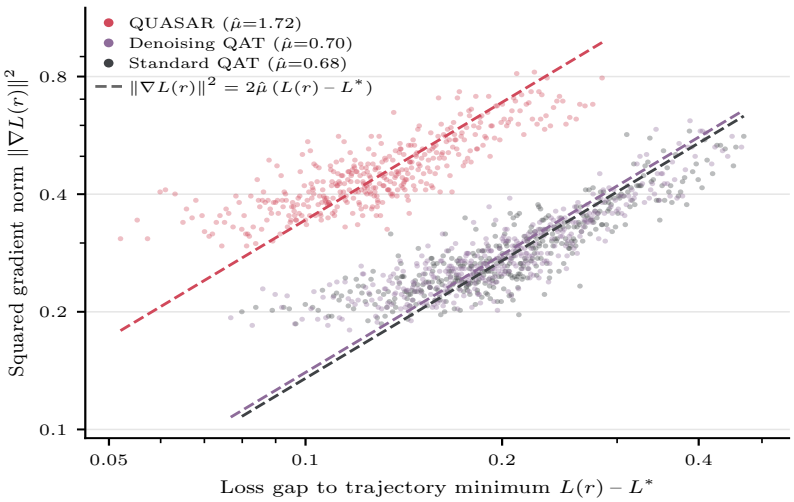


Figure 16: **Training trajectories support a PL relation.** Each point is one step of an INT2 Qwen3-4B-Thinking run, with loss and gradient measured at the reconstruction r. For all three methods, the squared gradient norm scales with the gap above the trajectory minimum. QUASAR's estimated $\hat{\mu} = 1.72$ is about $2.5\times$ the baseline values. In Theorem 1(ii), a larger μ gives faster geometric convergence and smaller floor terms, which scale with $1/\mu$ or $1/\mu^2$. This provides an optimization-side view of the healing curves (Figure 3): QUASAR's reconstruction adds less loss, so more of the remaining gap produces useful gradient, enabling faster healing and a lower loss floor.

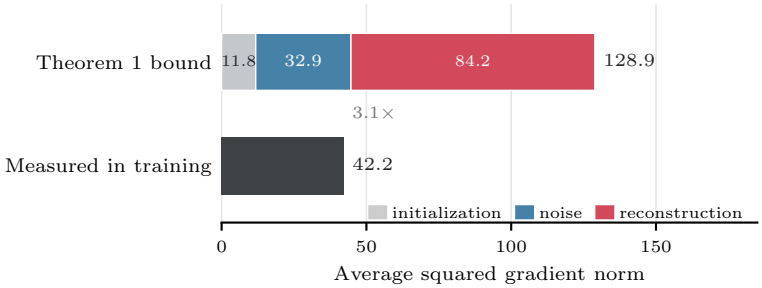


Figure 17: **Reconstruction error dominates the measured bound.** We evaluate each term of Theorem 1(i) on an INT2 QUASAR run with plain SGD and learning rate 10^{-3}. We measure $\lambda_{\max}$ by power iteration (approximating L), σ^2 from per-batch gradients, and C using Figure 14. The resulting bound is within $3.1\times$ of the observed average squared gradient norm, and reconstruction error is its largest term.

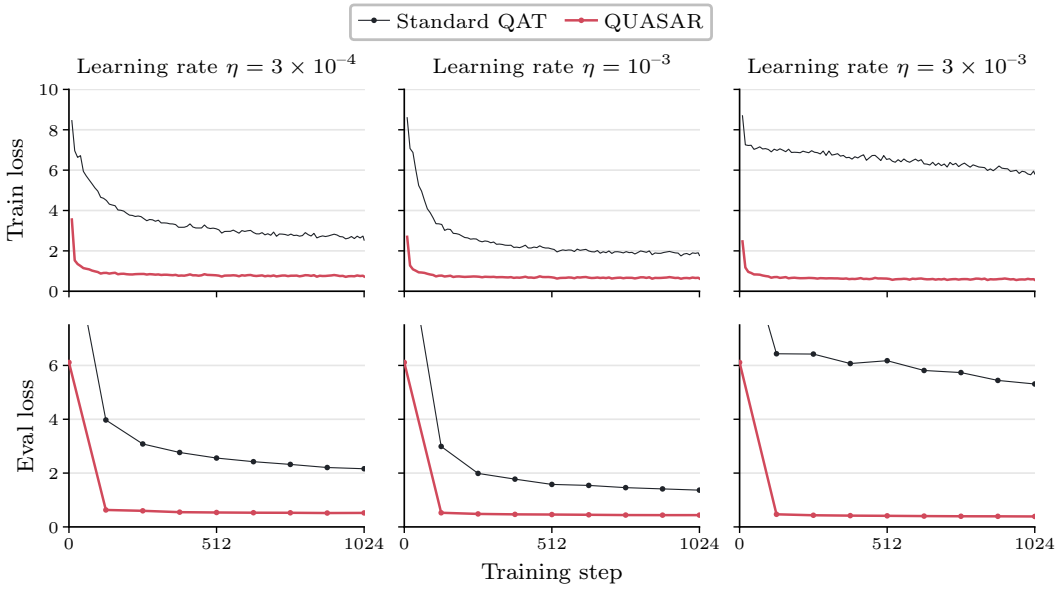


Figure 18: **QUASAR retains its advantage under plain SGD.** Training (top) and held-out (bottom) loss for Standard QAT and QUASAR at INT2 across three constant learning rates. QUASAR reaches lower loss floors at every stable learning rate.

D EXPERIMENTAL DETAILS

This appendix specifies the training and evaluation protocols for Section 5. Format-specific algorithms are given in Appendix A.1, and released checkpoints are listed in Appendix H.

Quantization scope. The controlled INT experiments use asymmetric weight-only INT4, INT3, or INT2 quantization with group size 128. We quantize every linear projection in the transformer blocks and keep activations, embeddings, the language-model head, and norms in BF16. Healing updates only the quantized linear weights; adaptation updates all student parameters.

Healing. We apply QAD for 4,096 steps at sequence length 4,096. Qwen3-4B uses global batch 32 and learning rates $2, 3, 5 \times 10^{-5}$ at INT4, INT3, and INT2. Llama-3.1-8B uses global batch 128 and learning rates $1, 2, 3 \times 10^{-5}$. This corresponds to roughly 60k supervised tokens per step. Training uses KL distillation from the full-precision teacher. QUASAR uses AdamW's second moment as saliency and searches clipping factors from 0.30 to 1.00 in increments of 0.05.

Adaptation. We train Qwen3-4B-Base on OpenMathReasoning for 4,096 steps at sequence length 18,432 and global batch 8. All methods use cross-entropy, AdamW with learning rate 5×10^{-5}, a cosine schedule with 1% warmup, no weight decay, and gradient clipping at 1. QUASAR uses the same integer quantizer and scale-search grid as in healing.

NVFP4. NVFP4 QAD uses AdamW with learning rate 10^{-6} and global batch 32. We materialize the final checkpoints and evaluate them through vLLM's native NVFP4 path (Kwon et al., 2023; vLLM Project, 2026b).

Baselines. Standard QAT uses fixed min–max ranges; LSQ (Esser et al., 2020) learns the scales; Denoising QAT (Ye et al., 2026) refits the dequantization parameters; BitDistiller (Du et al., 2024) combines clipping with self-distillation; RTN rounds to the nearest grid point; GPTQ (Frantar et al., 2023) minimizes a second-order reconstruction error; and AWQ (Lin et al., 2024) protects activation-salient weights.

Evaluation protocol. For healing, we report fidelity to the full-precision model and downstream accuracy. For adaptation, we report held-out perplexity and accuracy on five math benchmarks. NVFP4 results include fidelity, reasoning, coding, and long-context evaluations. Full task protocols are given with the corresponding results in Appendices E, F, and G.

E ADDITIONAL HEALING RESULTS

This appendix extends the healing results of Section 5.1 with additional fidelity and downstream metrics, training dynamics, and an LLM-judge analysis of INT2 response quality. It also reports representative generations and QUASAR ablations. NVFP4 results are in Appendix G.

E.1 STANDARD TASKS AND FIDELITY

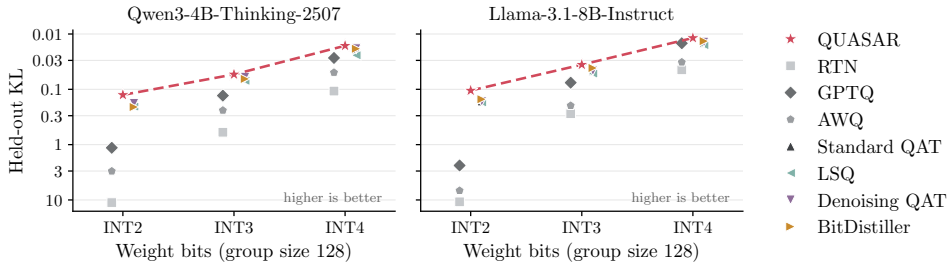


Figure 19: **QUASAR gives the lowest held-out KL across both models and all bit widths.** Forward KL to the full-precision model after QAD of Qwen3-4B-Thinking-2507 (left) and Llama-3.1-8B-Instruct (right). Lower is better.

Table 6: **QUASAR gives the highest fidelity in all six model–bit-width settings.** Held-out forward KL, top-1 agreement, and accuracy on eight benchmarks for QAD of Qwen3-4B-Thinking-2507 and Llama-3.1-8B-Instruct.

	Qwen3-4B-Thinking-2507											Llama-3.1-8B-Instruct									
Method	KL ↓	Top-1 ↑	GSM8K	MMLU	ARC-C	ARC-E	HellaSwag	WinoGrande	TruthfulQA	IFEval	**Avg.**	KL ↓	Top-1 ↑	GSM8K	MMLU	ARC-C	ARC-E	HellaSwag	WinoGrande	TruthfulQA	IFEval	**Avg.**
Teacher (FP16)	–	–	87.0	68.7	53.1	76.7	65.7	66.1	57.5	54.3	66.1	–	–	70.1	68.3	55.6	79.9	79.5	73.6	54.5	73.9	69.4
2-bit (INT2)																						
RTN	11.166	0.5	0.0	25.4	25.6	25.9	26.0	52.0	46.8	8.3	26.3	10.810	0.5	0.0	25.5	26.2	26.1	26.1	51.8	47.6	11.8	26.9
GPTQ	1.140	67.7	0.2	23.4	25.1	30.0	30.1	49.3	**52.9**	8.9	27.5	2.380	51.8	0.0	24.9	23.3	28.9	30.0	48.0	49.0	9.1	26.7
AWQ	3.016	37.0	0.0	23.4	23.2	31.6	29.1	51.1	52.8	9.6	27.6	6.769	12.9	0.0	23.6	23.3	25.9	27.2	48.2	48.8	7.9	25.6
Standard QAT	0.186	89.9	47.4	34.5	31.2	44.8	46.9	54.5	51.1	33.5	43.0	0.163	90.1	1.8	23.4	22.9	30.9	34.8	49.3	46.8	44.4	31.8
LSQ	0.205	89.2	42.6	34.6	30.6	44.9	47.6	55.2	47.2	35.1	42.2	0.172	89.6	42.2	39.7	32.8	52.8	55.7	53.5	47.4	42.1	45.8
Denoising QAT	0.179	90.1	45.6	36.0	30.1	47.9	46.7	56.1	49.9	34.9	43.4	0.165	90.1	0.0	24.6	26.2	25.6	26.9	50.0	49.1	40.9	30.4
BitDistiller	0.209	89.1	49.0	43.2	38.0	**60.1**	52.6	57.9	52.6	37.7	48.9	0.151	90.2	53.1	49.3	43.5	69.9	66.6	66.3	47.7	51.8	56.0
QUASAR (ours)	**0.126**	**92.0**	**68.8**	**48.9**	**38.8**	55.9	**56.4**	**59.2**	50.5	**47.5**	**53.2**	**0.105**	**92.0**	**66.4**	**52.5**	**43.8**	**73.1**	**68.1**	**66.8**	**49.5**	**56.0**	**59.5**
3-bit (INT3)																						
RTN	0.597	78.8	20.5	53.6	39.4	56.1	54.1	58.1	51.4	18.9	44.0	0.276	85.6	19.0	44.8	40.2	64.6	69.5	66.9	47.0	52.9	50.6
GPTQ	0.130	91.5	77.3	61.0	48.0	75.5	61.0	61.8	54.0	49.2	61.0	0.076	92.8	61.4	59.2	49.1	71.6	75.7	71.0	**53.1**	69.1	63.8
AWQ	0.240	88.1	73.2	60.1	46.4	72.3	60.7	62.7	49.2	41.8	58.3	0.197	88.0	23.9	53.2	43.0	67.6	72.2	67.6	42.1	58.6	53.5
Standard QAT	0.062	94.2	78.5	62.0	46.8	74.1	61.9	64.4	56.1	53.2	62.1	0.046	94.7	70.7	59.7	47.4	72.8	75.2	**72.9**	52.1	69.9	65.1
LSQ	0.070	93.8	**83.0**	62.3	**49.4**	**75.8**	62.7	**65.6**	55.4	52.9	**63.4**	0.052	94.2	**76.8**	62.3	51.8	76.9	75.8	71.6	50.9	67.5	66.7
Denoising QAT	0.060	94.3	80.4	62.7	46.2	73.5	61.6	64.2	56.1	51.0	62.0	0.046	94.7	68.8	59.8	48.0	74.5	75.0	72.2	51.0	70.2	64.9
BitDistiller	0.065	94.2	81.0	63.7	47.0	69.0	63.0	63.5	**56.3**	**53.4**	62.1	0.041	94.8	74.2	61.6	53.1	79.0	76.4	71.9	52.2	68.6	67.1
QUASAR (ours)	**0.054**	**94.9**	81.7	**64.5**	48.3	72.9	**63.2**	64.7	55.5	50.6	62.7	**0.036**	**95.2**	74.1	**63.3**	**54.9**	**79.1**	**77.1**	71.0	51.6	**71.5**	**67.9**
4-bit (INT4)																						
RTN	0.108	92.2	80.2	66.1	48.8	69.7	63.5	65.4	**58.4**	54.2	63.3	0.044	94.4	70.7	64.1	53.3	76.9	78.0	73.9	50.7	72.1	67.5
GPTQ	0.027	96.2	84.9	67.1	51.4	75.0	64.7	65.4	56.5	51.4	64.6	0.015	96.8	69.0	**67.3**	52.0	77.3	78.7	73.2	**55.1**	73.6	68.3
AWQ	0.050	94.7	84.6	67.6	51.5	74.9	**65.2**	**66.6**	56.7	56.6	65.5	0.032	95.2	65.6	66.6	53.7	78.3	78.7	72.4	53.7	73.4	67.8
Standard QAT	0.019	96.9	84.9	67.0	49.7	73.5	65.1	65.2	56.7	54.3	64.6	0.014	97.0	76.3	66.3	54.4	77.7	78.4	73.8	53.9	73.0	69.2
LSQ	0.024	96.4	86.0	66.8	50.6	75.0	**65.2**	65.6	57.1	56.4	65.3	0.016	96.8	73.2	66.5	54.6	**80.1**	78.7	74.1	53.2	**74.1**	69.3
Denoising QAT	0.018	96.9	85.0	67.1	49.2	72.9	64.7	65.7	57.3	**56.9**	64.9	0.014	97.0	**77.0**	66.2	**55.7**	77.8	78.6	73.9	53.6	72.1	69.3
BitDistiller	0.018	96.9	84.8	67.7	**52.6**	**76.0**	64.8	65.9	57.3	54.3	65.4	0.013	97.0	70.4	67.0	55.1	79.7	**78.8**	73.3	53.8	**74.1**	69.0
QUASAR (ours)	**0.016**	**97.1**	**86.4**	**68.0**	50.6	**76.0**	65.1	65.2	57.7	55.8	**65.6**	**0.012**	**97.3**	74.6	66.5	55.5	78.9	**78.8**	**74.2**	54.2	72.5	**69.4**

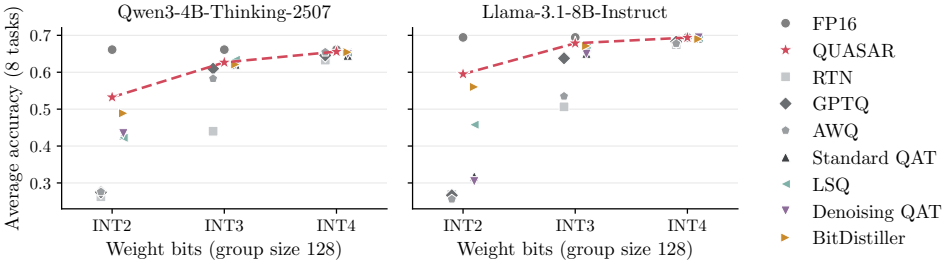


Figure 20: **QUASAR attains the highest INT2 average accuracy on both models.** Average accuracy over the eight benchmarks of Table 6 versus bit width for Qwen3-4B-Thinking-2507 (left) and Llama-3.1-8B-Instruct (right). Gray circles mark the full-precision models.

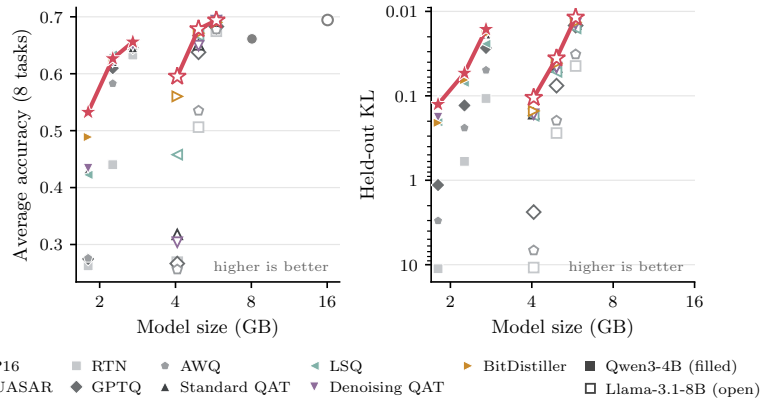


Figure 21: **QUASAR gives the best fidelity at each quantized model size.** Average accuracy over eight benchmarks (left) and held-out KL to the full-precision model (right) versus model size. Filled markers denote Qwen3-4B-Thinking-2507, open markers denote Llama-3.1-8B-Instruct, and gray circles mark the full-precision models.

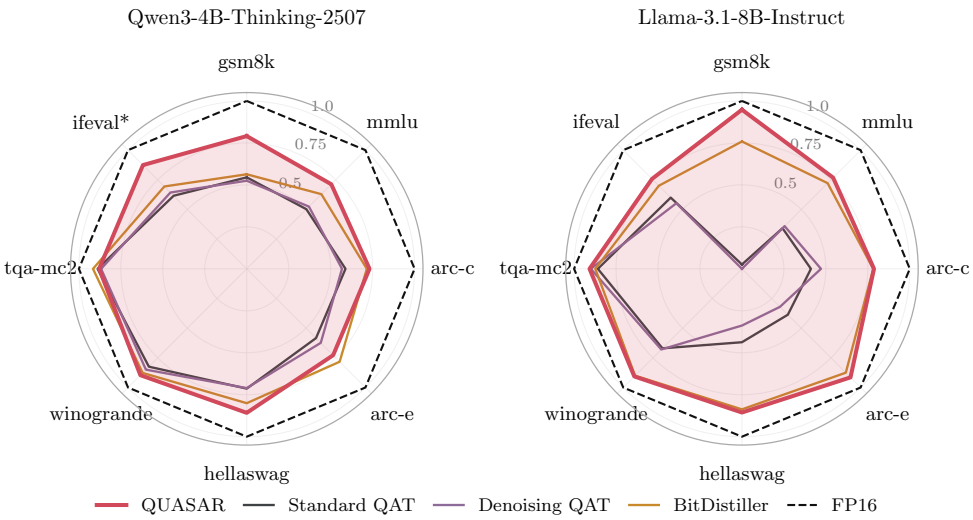


Figure 22: **QUASAR preserves INT2 accuracy broadly across tasks.** Per-task accuracy normalized by the full-precision model for Qwen3-4B-Thinking-2507 (left) and Llama-3.1-8B-Instruct (right) on the eight benchmarks of Table 6.

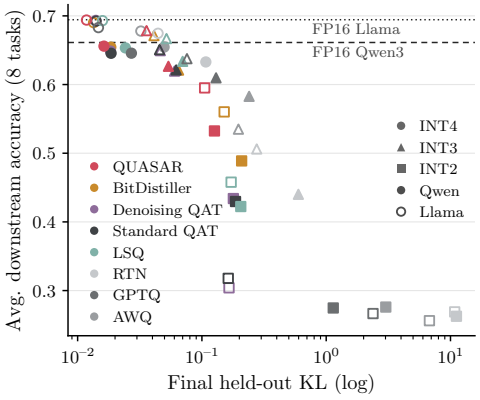


Figure 23: **Lower held-out KL correlates with higher downstream accuracy.** Average accuracy over eight benchmarks versus final forward KL for the QAT and PTQ methods of Table 6, across both models and all three bit widths. Dashed lines mark full-precision accuracy.

E.2 Training Dynamics

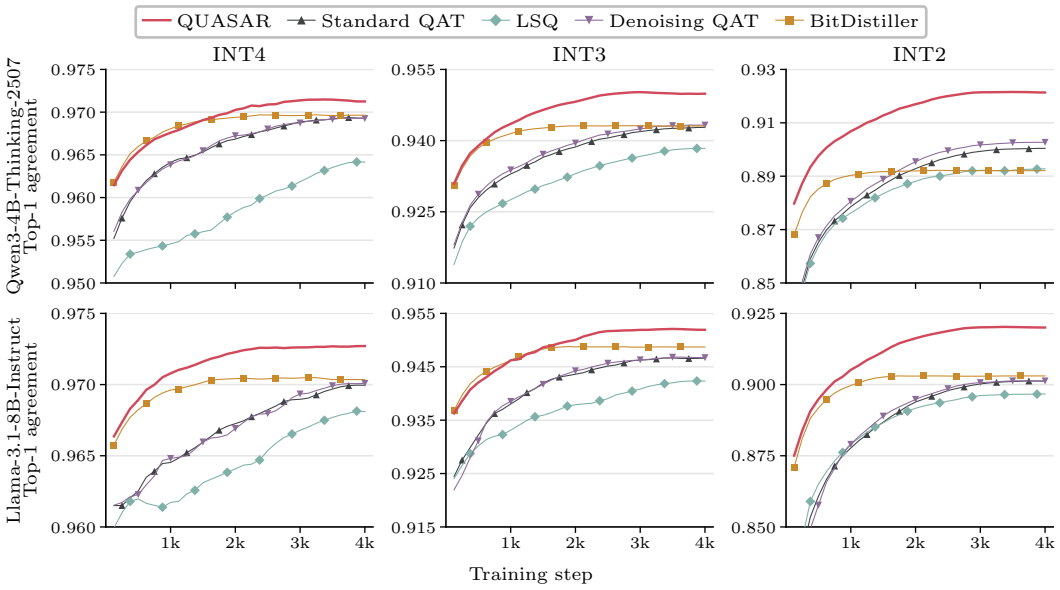


Figure 24: **QUASAR reaches the highest final top-1 agreement in all six settings.** Top-1 agreement with the full-precision model during QAD of Qwen3-4B-Thinking-2507 (top) and Llama-3.1-8B-Instruct (bottom) at INT4, INT3, and INT2.

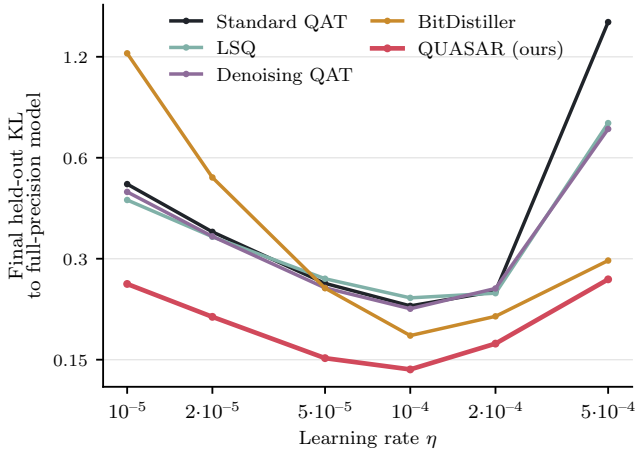


Figure 25: **QUASAR is robust to learning rate.** Final held-out KL after 1,000 steps of INT2 QAD of Qwen3-4B-Thinking-2507 across a 50× learning-rate range. QUASAR attains the lowest KL at every tested rate; Standard QAT diverges at high rates, while BitDistiller degrades at low rates.

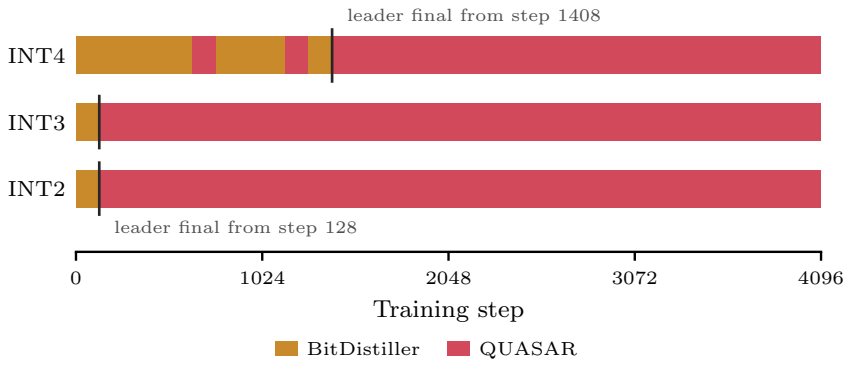


Figure 26: **QUASAR becomes the final leader early in training.** Each strip identifies the method with the lowest held-out KL among six methods for Qwen3-4B-Thinking-2507. QUASAR leads from step 128 at INT3 and INT2, and from step 1,408 at INT4, through the end of training.

E.3 LLM-AS-A-JUDGE PREFERENCE AT 2 BITS

Across 128 WildChat prompts per model, Llama-3.3-70B-Instruct prefers QUASAR to all seven quantized baselines for both models (Figure 27; Table 7). The margin is largest against PTQ, whose INT2 responses are often repetitive or incomplete (Figure 28). The full-precision models remain preferred. Figure 29 shows one representative response.

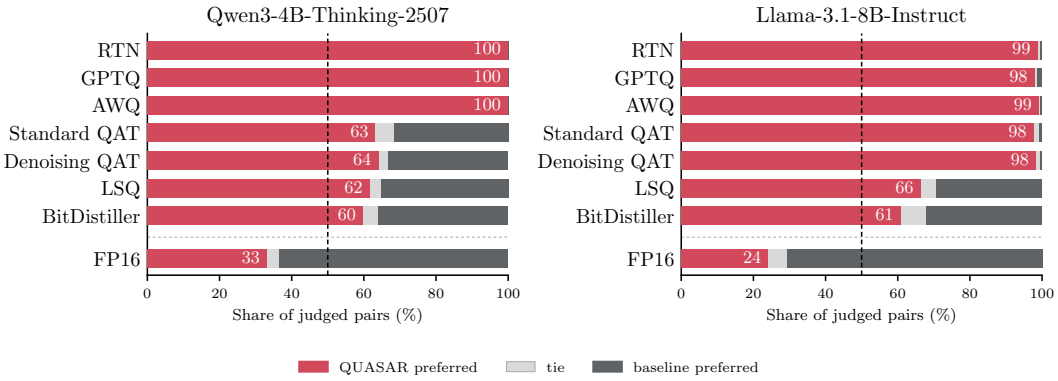


Figure 27: **The judge prefers QUASAR to every INT2 quantized baseline.** Pairwise win, tie, and loss rates on responses to 128 WildChat prompts for Qwen3-4B-Thinking-2507 and Llama-3.1-8B-Instruct. Llama-3.3-70B-Instruct serves as the judge.

Table 7: **QUASAR wins all 14 comparisons with quantized baselines.** Mean per-prompt preference in $[-1, 1]$, with 95% bootstrap confidence intervals over 128 prompts. Positive values favor QUASAR. The final row compares QUASAR with the full-precision model.

vs. QUASAR	Qwen3-4B-Thinking-2507	Llama-3.1-8B-Instruct
RTN	$+1.00$ $[+1.00, +1.00]$	$+0.98$ $[+0.96, +1.00]$
GPTQ	$+1.00$ $[+1.00, +1.00]$	$+0.97$ $[+0.94, +0.99]$
AWQ	$+1.00$ $[+1.00, +1.00]$	$+0.99$ $[+0.97, +1.00]$
Standard QAT	$+0.32$ $[+0.21, +0.42]$	$+0.97$ $[+0.94, +0.99]$
LSQ	$+0.27$ $[+0.15, +0.38]$	$+0.37$ $[+0.25, +0.49]$
Denoising QAT	$+0.31$ $[+0.21, +0.41]$	$+0.98$ $[+0.95, +1.00]$
BitDistiller	$+0.24$ $[+0.12, +0.35]$	$+0.29$ $[+0.17, +0.41]$
FP16	-0.30 $[-0.40, -0.20]$	-0.46 $[-0.56, -0.37]$

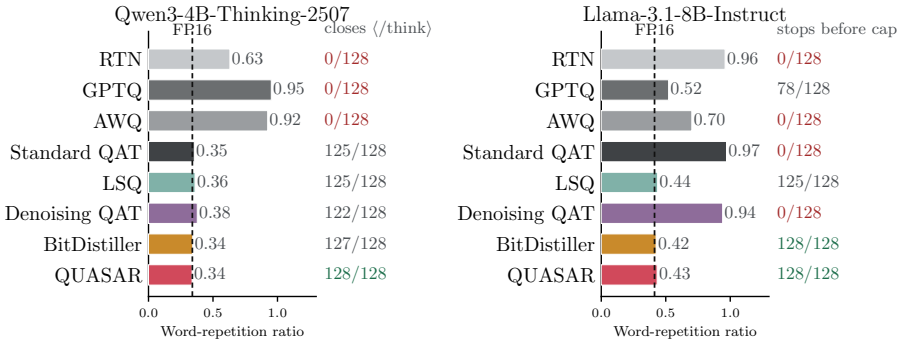


Figure 28: **QUASAR matches the full-precision models' response stability at INT2.** Mean word-repetition ratio over 128 WildChat prompts for Qwen3-4B-Thinking-2507 (left) and Llama-3.1-8B-Instruct (right), with greedy decoding. Dashed lines mark the full-precision models; counts show responses that close the reasoning block for Qwen or stop before the token cap for Llama.

Prompt: *Write a story about a man being stranded on an island.*	
FP16	The salt stung Leo's eyes before he even registered the crash. One moment, he was shouting into the cockpit of his battered sedan, the next, the world …
2-bit (INT2)	
RTN	max o mil MD o. over md edge md mdish ai } ats } } further display ○scs <igrated found "arql eventually o, same, if (cs1) < <○ <ql <, \ql o < <! (< …
GPTQ	*A [R] :Help "l" (or: "a" in 1000+). 0 * F : 0 (To the 0) : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 10000: 0 : 10000: 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : …
AWQ	go {r=0 {r=1, 100 100 100 100○REEN○.1 ○○100○○○ 100○ 10 100. 100 10 10 1 10 1 1 1 0 1 100 1 1 1 1 1 (1 ○ 1 2 1 02 1 1 1 1 1 1 1. 1 1 1 1 1 1 1 1 1 1 1 …
Standard QAT	The first thing Elias noticed wasn't the ocean, but the *absence* of it. Not the roar of waves, but the silence of his own thoughts, a hollow space wh …
Denoising QAT	Leo woke to the sound of water. Not the gentle lapping of a tide, but a relentless, crashing roar against the rocks of a tiny, rocky island he'd never …
LSQ	Leo woke to the smell of salt and wet earth. The first thing he noticed wasn't the storm, but the *absence* of sound. The roar of the ocean that had b …
BitDistiller	Kai's first thought wasn't panic, but a slow, heavy dread. The salt air stung his eyes, thick with the scent of wet earth and something else—something …
QUASAR (ours)	The last thing Alex remembered was the *thud* of the car door slamming shut, the roar of the storm tearing through the windshield, and the sickening l …
3-bit (INT3)	
RTN	Okay, the user wants a story about a man stranded on an island. Hmm, let's unpack that. First, I should consider what they might need. Are they writin …
GPTQ	The sea had taken him. Not with a crash, but with a slow, suffocating pull. Elias had been a fisherman, not a sailor, and the *Mariana*—his small, stu …
AWQ	The storm had been a fury of glass and thunder, tearing the sea from its bones. Elias woke to the *crack* of splintered wood and the sharp, metallic s …
Standard QAT	Elias woke to the sound of the ocean, not the crash of waves, but the *thump-thump-thump* of his own heart against his ribs. Salt stung his eyes, and …
Denoising QAT	Leo woke to the smell of salt and something else – wet earth, maybe, or decaying leaves. His head throbbed, a dull ache behind his eyes. He was lying …
LSQ	## The Island That Forgot to Breathe The last thing Elias remembered was the *thump* of the boat hitting the water, the sickening lurch of the world s …
BitDistiller	Leo's last memory was the *thwack* of a wave against the hull of the *Marlin*, the sharp sting of salt in his eyes, and the terrifying, silent *thud* …
QUASAR (ours)	Leo's last memory was the *scream* of the engine, the sickening lurch as the boat plunged into the churning grey sea, and the world dissolving into a …
4-bit (INT4)	
RTN	Leo didn't know how he got there. One moment he was a man in a sleek, unremarkable boat, the next—*thud*—a violent wave slammed the hull, the engine d …
GPTQ	Elias's world dissolved in a roar. One moment, he was gripping the steering wheel of his aging sedan, the rain lashing the windshield like a thousand …
AWQ	Leo's last thought before the world dissolved into water and silence was the *thump* of the boat's hull against the rocks. Then, the roar of the ocean …
Standard QAT	Leo didn't know how long he'd been adrift when the first wave of panic hit. It wasn't the cold, the salt, or the crushing weight of the ocean's indiff …
Denoising QAT	Leo woke to the sound of the ocean, not the roar of a storm, but the low, insistent *thrum* of the sea against the rocks. His head throbbed, a dull ac …
LSQ	Leo's last memory was the *thud* of the plane hitting the water, not the roar of the engines or the frantic shouts of the crew. He woke up gasping, th …
BitDistiller	The crash wasn't loud. It was a sickening *thud* of metal and earth, followed by a silence so profound it felt like the island itself had held its bre …
QUASAR (ours)	The impact wasn't a crash. It was a *sigh* of the world ending. Ben's world dissolved into a sickening lurch, the roar of the ocean swallowing his scr …

Figure 29: The beginning of each method's response to one held-out prompt, for Qwen3-4B-Thinking-2507 at INT2, INT3, and INT4 with greedy decoding.

E.4 ABLATIONS

Table 8: Ablation Study: Effect of QUASAR's scale search and weighted dequantization fit on Qwen3-4B-Thinking-2507 with INT2 QAD. We report final held-out KL and its change from Standard QAT.

Variant	Standard QAT	Adam fit	Init. only	Uniform fit	Fisher fit	**QUASAR** (ours)
Scale search	none	none	at init.	every step	every step	every step
Saliency	none	Adam 2nd moment	uniform	uniform	Fisher	Adam 2nd moment
KL ↓	0.175	0.164	0.172	0.114	<u>0.113</u>	**0.112**
Δ vs. Standard QAT	–	-6.1%	-1.6%	-34.7%	<u>-35.5%</u>	$\mathbf{-35.9}\%$

F Additional Adaptation Results

This appendix extends the adaptation results of Section 5.1 with training curves and three analyses of the INT2 Qwen3-4B-Base checkpoints: answer outcomes, variation across repeated samples, and divergence from FP-SFT along reasoning traces.

QUASAR's advantage is largest on benchmarks requiring long reasoning. FP-SFT generates median responses of 27k tokens on AIME'24 and 20k on HMMT'25. At INT2, QUASAR is the only quantized model to score above zero on HMMT'25, suggesting that it better preserves the adapted behavior over long generations.

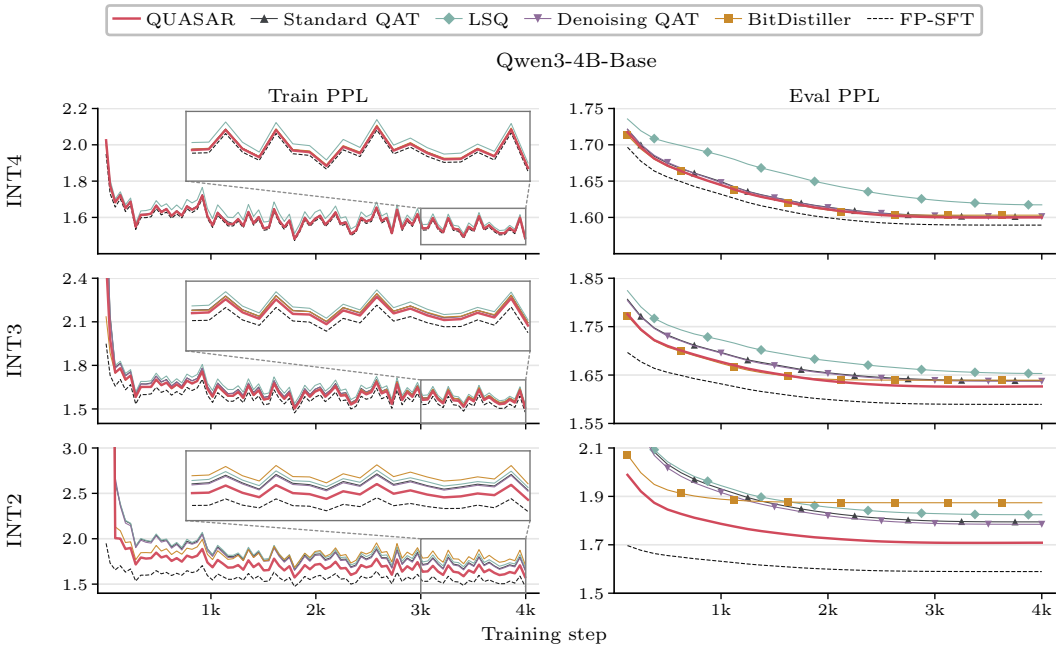


Figure 30: **QUASAR reaches the lowest held-out perplexity at every bit width.** Training perplexity over the full run (left) and final 1,000 steps (center), and held-out perplexity (right), for QAT of Qwen3-4B-Base on OpenMathReasoning. FP-SFT is included as a full-precision reference.

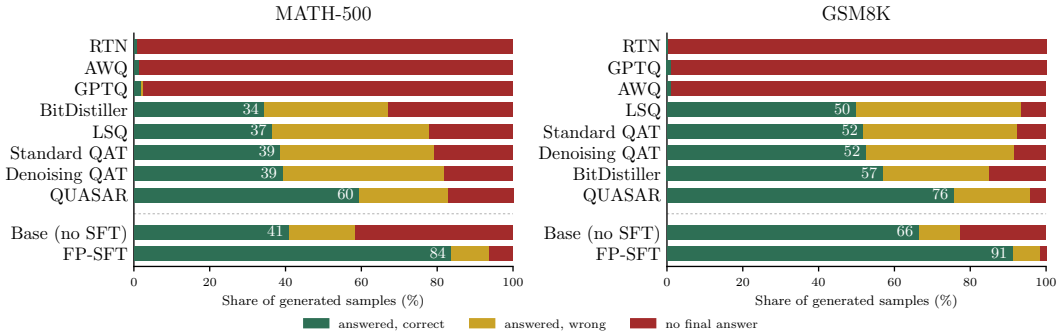


Figure 31: **Answer outcomes for the INT2 checkpoints.** Share of generated samples that produce a correct answer, an incorrect answer, or no final answer on MATH-500 and GSM8K. The base model and FP-SFT are included as references.

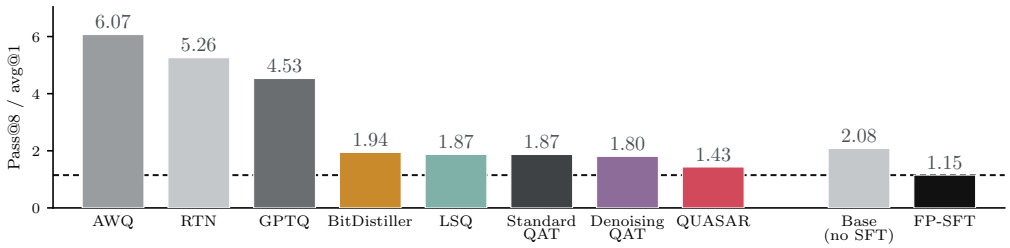


Figure 32: **Variation across repeated INT2 samples.** Ratio of pass@8 to avg@1 on MATH-500 for Qwen3-4B-Base checkpoints, using eight samples per problem at temperature 0.6. A ratio of 1 means that the same problems are solved on every attempt; higher values indicate less consistent successes. The base model and FP-SFT are included as references.

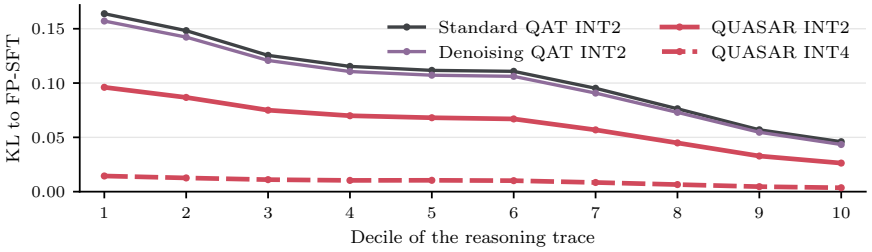


Figure 33: **Divergence from FP-SFT along reasoning traces.** Forward KL to FP-SFT by trace decile for selected quantized checkpoints, averaged over 1,000 MATH-500 traces generated by FP-SFT. Every checkpoint is evaluated on the same token sequences.

G ADDITIONAL NVFP4 RESULTS

This appendix collects the training curves and the full evaluation results for the NVFP4 experiments of Section 5.2.

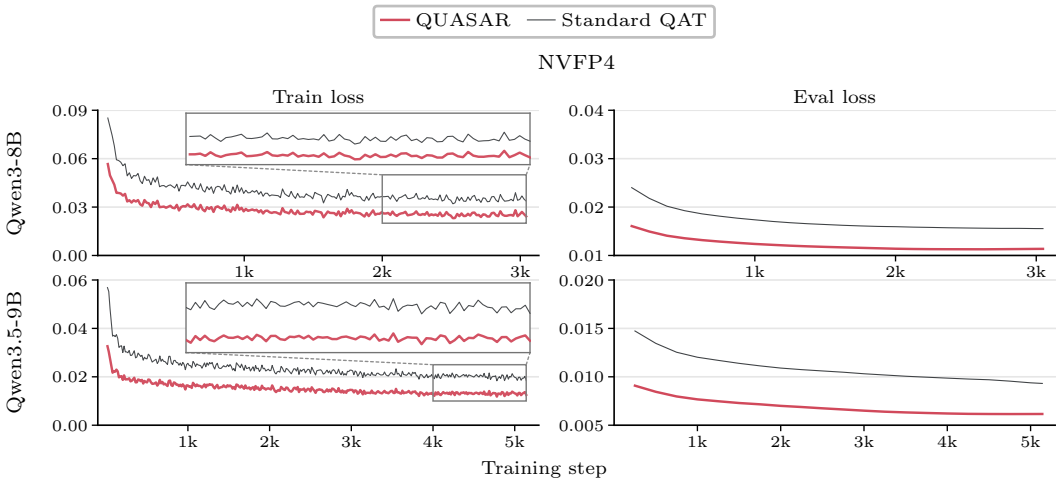


Figure 34: **QUASAR reaches lower held-out loss for both NVFP4 models.** Training forward KL over the full run (left) and final 1,000 steps (center), and held-out forward KL (right), for NVFP4 QAD of Qwen3-8B (top) and Qwen3.5-9B (bottom).

Table 9: **QUASAR gives the best average accuracy among the quantized checkpoints.** Accuracy on nine downstream benchmarks for Qwen3-8B and Qwen3.5-9B. The materialized NVFP4 checkpoints are evaluated through vLLM's native W4A4 path.

	Downstream task accuracy (%) ↑									
Method	**Avg.**	GSM8K	MMLU	ARC-C	ARC-E	Hella.	WinoG.	TQA	IFEval	GPQA-D
Qwen3-8B										
BF16	68.6	86.7	72.9	56.7	80.9	75.0	67.7	54.4	84.8	38.4
Standard QAT	67.1	83.9	70.8	55.4	80.3	**73.6**	**67.1**	53.9	80.6	37.9
QUASAR (ours)	**67.6**	**87.0**	**71.1**	**55.7**	**80.4**	**73.6**	66.8	**54.2**	**81.2**	**38.4**
Qwen3.5-9B										
BF16	71.2	74.4	78.6	55.7	74.2	78.1	72.8	53.7	74.4	79.1
Standard QAT	69.4	73.3	76.7	56.3	**75.6**	76.7	70.8	52.8	66.0	76.1
QUASAR (ours)	**70.2**	**73.7**	**77.0**	**56.6**	**75.6**	**77.2**	**70.9**	**53.9**	**68.5**	**78.1**

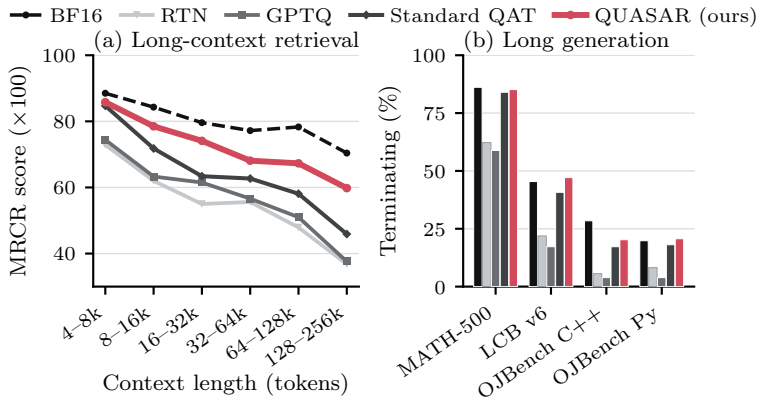


Figure 35: **QUASAR better preserves long-horizon behavior in NVFP4.** Expanded Qwen3.5-9B results from Table 3: (a) OpenAI-MRCR score by context length; (b) share of long generations that terminate within the 32k-token output budget. Higher is better.

Table 10: **QUASAR has the lowest or tied-lowest non-termination rate among the quantized checkpoints.** Percentage of generations that reach the 32,768-token output budget without stopping, using the same generations as Table 3. Lower is better.

Method	MATH-500	LiveCodeBench v6	OJBench (C++)	OJBench (Python)
Qwen3-8B				
BF16	0.5	13.6	1.3	2.2
RTN	1.4	40.8	20.3	26.7
GPTQ	**0.5**	24.3	11.6	10.3
Standard QAT	**0.5**	21.6	4.7	5.6
QUASAR (ours)	**0.5**	**18.4**	**3.0**	**2.2**
Qwen3.5-9B				
BF16	13.9	54.6	71.5	80.2
RTN	37.8	78.1	94.4	91.8
GPTQ	41.1	82.8	96.1	96.1
Standard QAT	16.0	59.2	82.8	81.9
QUASAR (ours)	**14.8**	**52.9**	**79.7**	**79.3**

H Comparison with Google's QAT and Other 4-bit Checkpoints

Table 11 extends Table 3 with Qwen3.5-4B and Gemma-4 12B. The RTN rows, and the GPTQ rows at NVFP4 W4A16 and INT4 g64, are our own conversions of the BF16 models; all other rows are public artifacts, listed in Table 12.

Table 11: Comparison with released 4-bit checkpoints across four model families. We report fidelity to BF16 and accuracy on each model's benchmark suite. GGUF KL values are comparable only within each block. ‡ denotes mixed NVFP4, FP8, and BF16 precision.

Model	Format	bpw	Fidelity		Accuracy (%) ↑							
			KL ↓	Top-1 ↑	GSM8K-P	MMLU-P	IFEval	MATH	AIME25	GPQA-D		**Avg.**
Qwen3.5-4B												
BF16	BF16	16	–	–	94.0	78.9	88.0	84.4	79.2	77.3		83.6
RTN	NVFP4 W4A16	4.50	.042	93.8	94.9	77.5	86.3	83.8	67.1	74.1		80.6
GPTQ	NVFP4 W4A16	4.50	.023	95.7	**95.1**	75.2	84.5	83.2	52.9	69.9		76.8
GPTQ	INT4 g128	7.49	.049	93.1	94.0	77.5	87.1	83.0	65.0	71.2		79.6
AWQ	INT4 g32	4.50	.042	93.9	93.9	77.6	87.6	84.2	72.5	74.6		81.7
ModelOpt PTQ	NVFP4 W4A4	4.50	.043	93.7	94.2	77.9	**88.2**	**84.6**	65.8	73.6		80.7
QUASAR (ours)	NVFP4 W4A16	4.50	**.018**	**96.2**	94.7	**78.5**	87.2	83.8	**74.2**	**77.9**		**82.7**
GGUF (llama.cpp)												
RTN	GGUF Q4_0	4.50	.372	89.0	94.3	77.2	90.6	84.2	64.2	73.7		80.7
KD-QAT g64	GGUF Q4_0	4.50	.363	88.4	**94.9**	77.4	84.5	84.4	64.6	73.9		80.0
imatrix (PTQ)	GGUF Q4_0	4.84	.284	90.4	94.3	78.0	90.6	**86.0**	70.8	**76.1**		82.6
QUASAR (ours)	GGUF Q4_0	4.50	**.235**	**91.3**	**94.9**	**78.2**	**90.9**	83.8	**77.9**	74.6		**83.4**

			KL ↓	Top-1 ↑	GPQA-D	MMLU-P	AIME25	LCB	RUL32K	RUL64K	RUL128K	**Avg.**
Muse-Glimmer-30B												
BF16	BF16	16	–	–	62.1	83.1	86.7	53.4	86.6	85.2	81.0	76.9
RTN	NVFP4 W4A16	4.50	.054	93.2	**63.6**	82.9	83.3	48.1	84.5	84.0	83.6	75.7
GPTQ	NVFP4 W4A4	4.50	.066	92.4	63.1	83.0	86.7	**54.2**	86.2	82.4	80.6	76.6
AutoQuantize	NVFP4+FP8‡	5.52	.030	94.9	63.1	82.9	83.3	52.7	85.6	84.8	83.5	76.6
QUASAR (ours)	NVFP4 W4A16	4.50	**.018**	**96.1**	**63.6**	**83.3**	**90.0**	49.6	**87.2**	**86.4**	**84.1**	**77.7**
QUASAR (ours)	NVFP4 W4A4	4.50	.053	93.2	63.1	82.7	83.3	52.7	**87.2**	**86.4**	83.5	77.0

			KL ↓	Top-1 ↑	MMLU	IFEval	ARC-C	AGIEval	MMLU-P	TQA	MATH-h	**Avg.**
Gemma-4 12B												
BF16	BF16	16	–	–	72.8	88.0	50.2	24.3	49.9	62.1	74.7	60.3
RTN	INT4 g64	4.25	.194	89.2	60.6	79.7	46.4	19.4	39.0	60.6	50.5	50.9
GPTQ	INT4 g64	4.25	.041	95.2	68.5	86.9	47.8	22.7	**48.2**	**62.1**	70.9	58.2
Google QAT	INT4 g32	4.50	.044	94.8	67.8	87.2	50.1	22.6	47.9	60.0	**73.2**	58.4
QUASAR (ours)	INT4 g64	4.25	**.034**	**95.5**	**70.4**	**87.6**	**51.3**	**24.2**	45.8	59.2	71.9	**58.6**
GGUF (llama.cpp)												
RTN	GGUF Q4_0	4.50	.156	89.8	59.2	87.4	47.8	22.7	38.4	60.2	62.3	54.0
Google QAT	GGUF Q4_0	4.50	.069	93.5	**71.0**	**89.1**	48.9	22.8	**45.6**	**61.3**	**74.8**	**59.1**
imatrix (PTQ)	GGUF Q4_0	4.52	.108	91.6	64.9	87.1	48.6	23.5	43.4	60.3	66.8	56.4
QUASAR (ours)	GGUF Q4_0	4.50	**.048**	**94.3**	70.3	87.6	**51.5**	**24.2**	45.5	59.1	72.5	58.7

			KL ↓	Top-1 ↑	TrvQA	NQ	TQA	ARC-E	ARC-C	GSM8K	IFEval	MATH-h	**Avg.**
Gemma-4 E4B													
BF16	BF16	16	–	–	23.6	3.9	59.1	79.7	56.5	84.1	85.2	63.4	56.9
RTN	INT4 g64	4.25	.079	91.5	**25.2**	3.0	**59.3**	78.3	55.2	79.7	83.0	56.1	55.0
GPTQ	INT4 g64	4.25	.031	94.7	21.3	3.3	56.1	78.2	54.9	**83.5**	**83.4**	59.4	55.0
Google QAT	INT4 g32	4.50	.064	92.6	20.1	2.4	55.4	78.3	56.0	81.5	81.9	59.4	54.4
QUASAR (ours)	INT4 g64	4.25	**.022**	**95.5**	23.8	**4.7**	58.2	**79.8**	**56.2**	82.0	82.6	**60.7**	**56.0**
GGUF (llama.cpp)													
RTN	GGUF Q4_0	4.50	.194	85.7	17.3	2.0	58.5	76.2	53.8	75.8	80.2	48.7	51.6
Google QAT	GGUF Q4_0	4.50	.088	90.6	18.5	2.3	55.5	78.7	**56.1**	81.4	81.7	58.7	54.1
Google QAT repack	GGUF Q4_0	4.50	.071	91.8	20.0	2.2	56.2	79.1	55.4	**83.5**	**83.2**	**61.8**	55.2
imatrix (PTQ)	GGUF Q4_0	4.52	.067	91.1	20.6	3.0	**58.8**	79.3	53.8	80.4	**83.2**	59.7	54.9
QUASAR (ours)	GGUF Q4_0	4.50	**.044**	**92.8**	**24.0**	**4.7**	58.2	**79.7**	**56.1**	81.8	82.4	60.5	**55.9**

bpw includes the quantized transformer-block weights, scales, and zero-points. Embeddings, output layers, norms, and small gating tensors are excluded.

Artifact definitions. For rows not described above:

- **GPTQ (INT4 g128), AWQ (INT4 g32), and GPTQ (NVFP4 W4A4)** are checkpoints created with LLM Compressor (vLLM project, 2024); gk denotes one scale per k weights, and W4A4 also quantizes activations to NVFP4.
- **ModelOpt PTQ** applies calibrated NVFP4 rounding without training. **AutoQuantize** selects NVFP4, FP8, or BF16 per layer under a 5.5-bit budget.
- **Google QAT** uses Google's unpublished INT4 recipe with 32-weight groups. Its Q4_0 variants convert or repack those weights for `llama.cpp`.
- **KD-QAT g64** distills Qwen3.5-4B on a 64-weight INT4 grid before conversion to 32-weight Q4_0 blocks.
- **imatrix (PTQ)** selects GGUF Q4_0 scales using activation-weighted reconstruction error; the GGUF **RTN** rows use plain round-to-nearest.

Table 12: Released checkpoints used in Tables 3 and 11.

Model	Row	Artifact
Qwen3.5-4B	BF16	`Qwen/Qwen3.5-4B` (Qwen Team, 2026a)
	GPTQ (INT4 g128)	`RedHatAI/Qwen3.5-4B-quantized.w4a16` (Red Hat AI, 2026b)
	AWQ (INT4 g32)	`cyankiwi/Qwen3.5-4B-AWQ-4bit` (cyankiwi, 2026)
	ModelOpt PTQ	`cosmicproc/Qwen3.5-4B-NVFP4` (cosmicproc, 2026)
	imatrix (GGUF)	`bartowski/Qwen_Qwen3.5-4B-GGUF` (bartowski, 2026)
	KD-QAT g64 (GGUF)	`YoozLabs/Qwen3.5-4B-qat-GGUF` (YoozLabs, 2026)
Muse-Glimmer-30B	BF16	`meta-models/Muse-Glimmer-30B` (Meta Superintelligence Lab, 2026)
	AutoQuantize	`nvidia/Muse-Glimmer-30B-NVFP4` (NVIDIA, 2026)
	GPTQ (NVFP4 W4A4)	`RedHatAI/Muse-Glimmer-30B-NVFP4` (Red Hat AI, 2026a)
Gemma-4 E4B	BF16	`google/gemma-4-E4B-it` (Google, 2026b)
	Google QAT	`google/gemma-4-E4B-it-qat-w4a16-ct` (Google, 2026e)
	Google QAT (GGUF)	`google/gemma-4-E4B-it-qat-q4_0-gguf` (Google, 2026f)
	Google QAT repack (GGUF)	`unsloth/gemma-4-E4B-it-qat-GGUF` (Unsloth, 2026d)
	imatrix (GGUF)	`unsloth/gemma-4-E4B-it-GGUF` (Unsloth, 2026c)
Gemma-4 12B	BF16	`google/gemma-4-12B-it` (Google, 2026a)
	Google QAT	`google/gemma-4-12B-it-qat-w4a16-ct` (Google, 2026c)
	Google QAT (GGUF)	`google/gemma-4-12B-it-qat-q4_0-gguf` (Google, 2026d)
	Google QAT repack (GGUF)	`unsloth/gemma-4-12B-it-qat-GGUF` (Unsloth, 2026b)
	imatrix (GGUF)	`unsloth/gemma-4-12b-it-GGUF` (Unsloth, 2026a)