CURVATURE-CORRECTED ACTIVATION GRADIENTS FOR BINARIZED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Binarized Neural Network (BNNs) training typically relies on the straight-through estimator (STE) as a practical approximation of the derivative of the quantization layer applied to the model weights and activations. We identify that the activation gradients in standard STE are biased and that this bias can be decomposed into two components: one related directly to the STE and another related to applying the classical chain rule through a discontinuous quantizer, which propagates the tangent of the downstream loss at the selected quantization level. In contrast, the distributional derivative of the population loss is governed by the finite loss jump between adjacent levels. A second-order expansion of this jump yields a gradient-correction term involving the quantization residual and the diagonal curvature of the downstream loss. We turn this result into a practical backward rule for activation quantizers by approximating the curvature component with an empirical-Fisher proxy and apply the proposed correction during the final 10% of training. We evaluate our method on binarized Llama-style language models trained with recent state-of-the-art methods, including QuEST and CAGE, and observe a 1.0–3.2% perplexity reduction across model sizes up to 1.6B parameters.

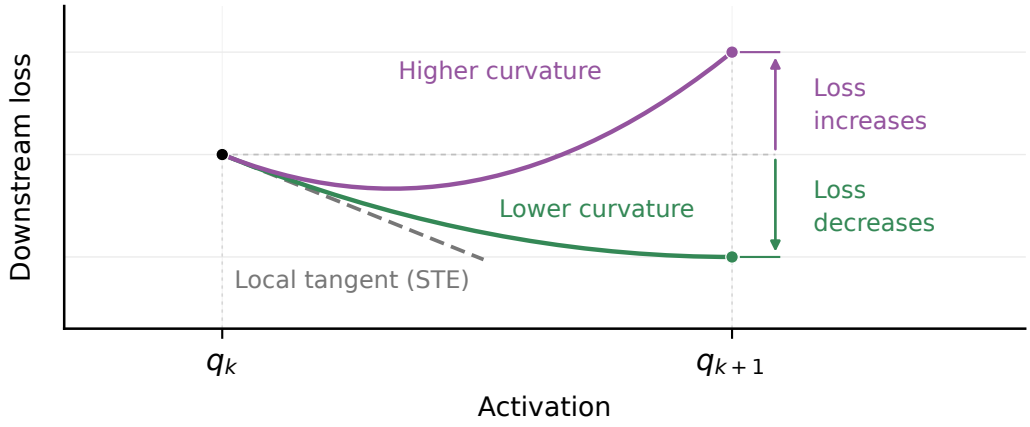


Figure 1: Why curvature matters in quantized training. Both curves have the same slope at quantization level q_k, so STE gives the same update direction. Yet switching to the adjacent level q_{k+1} lowers one loss and raises the other. Curvature helps estimate this loss jump by accounting for how the loss bends between quantization levels, beyond what the local slope alone reveals.

1 INTRODUCTION

Neural network quantization is an effective approach to reducing the memory and computational cost of deployed models. An extensive research effort has improved the accuracy of quantized networks since the early days of modern deep learning (Hubara et al., 2016; Choi et al., 2018; Esser et al., 2020). This effort has become especially important as large language models (LLMs) have rapidly grown in size and memory requirements (Frantar et al., 2023; Lin et al., 2026; Panferov et al., 2025;

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.

Tabesh et al., 2025). While post-training quantization compresses a trained model, quantization-aware training (QAT) evaluates the quantized model in the forward pass while updating latent high-precision parameters. Because a hard quantizer has zero classical derivative almost everywhere, QAT typically uses a surrogate derivative in the backward pass (Bengio et al., 2013).

Recent QAT methods have advanced the loss–model-size Pareto frontier for LLMs. In QuEST's scaling-law comparison, models with 4-bit weights and activations (W4A4) Pareto-dominate BF16 models, achieving lower validation loss with less model storage (Panferov et al., 2025). QuEST also enables stable training of models with 1-bit activations and weights (A1W1), but these models remain suboptimal in the same comparison. For weight quantization, ParetoQ likewise reports that binary models are generally less competitive than ternary, 2-bit, and 3-bit alternatives (Liu et al., 2025).

To improve training at such extremely low precisions, prior work has addressed both the quantization error in the forward pass, and the surrogate gradients used in backpropagation. QuEST combines distribution fitting to reduce quantization error in the forward pass with an error-based trust estimator (Panferov et al., 2025). CAGE derives a curvature-aware correction to latent-weight updates from a multi-objective formulation that balances task loss and quantization error (Tabesh et al., 2025). Theoretical analysis has characterized when the straight-through estimator (STE) provides a descent direction (Yin et al., 2019). A few other studies have proposed different ways to modify backward gradients in quantized networks (Lee et al., 2021; Zhao, 2025; Huang et al., 2026). These studies establish gradient estimation as a distinct design problem from choosing the forward quantizer. More broadly, recent studies of fully quantized LLM training also highlight the importance of avoiding bias when quantizing backward gradients (Chmiel et al., 2025; Panferov et al., 2026).

Nevertheless, much of recent QAT research focuses on weight quantization or latent-weight updates, while quantized activation gradients have received less attention in low-bit LLM training. We focus on the gradient propagated through quantized activations. Although a hard quantizer has zero classical derivative almost everywhere, the quantized *population* loss admits an analytically defined gradient, under regularity conditions. We derive the bias of the standard QAT activation gradient relative to this population gradient. This bias separates into an error from approximating the quantizer derivative and a "chain-rule error" stemming from using the downstream derivative at the selected quantization level instead of a coefficient determined by the finite loss jump across the threshold. We propose a second-order curvature correction to reduce this chain-rule error. Figure 1 illustrates why this correction can help: two losses with the same local derivative at one quantization level receive the same QAT coefficient for a fixed surrogate, yet their different curvature produces opposite loss jumps to the adjacent level. In the scalar analysis, this correction reduces the leading chain-rule term while keeping the quantizer-derivative surrogate fixed.

In practice, we use an empirical-Fisher-style curvature proxy and a bounded quantization residual to implement the correction without computing second derivatives of the loss with respect to activations. Added to QuEST and CAGE during the final 10% of training of A1W1 Llama-style models on C4, it reduces final validation perplexity by 1.0–3.2% across seven model sizes from 30M to 1.6B parameters (Table 1).

2 METHOD

We study the gradient propagated through a quantized activation. Standard QAT backpropagates the derivative of the downstream loss with respect to the quantized activation, multiplied by a straight-through estimator (STE) of the quantizer. We show that the distributional derivative instead requires the finite loss jump across the quantization threshold. Our method keeps the baseline STE and corrects its tangent coefficient toward this finite jump.

2.1 SETTING

Let $x \sim \mathcal{D}$, let $z := g_\theta(x) \in \mathbb{R}$ be a smooth pre-quantized activation, and let $Q(z) = \operatorname{sign}(z) \in \{-1, +1\}$. With activation scale $s > 0$ and loss f, the population objective is

$$\mathcal{L}(\theta) = \mathbb{E}_x[f(sQ(g_\theta(x)))]\,. \tag{1}$$

For the derivations, write $q := Q(z)$. Let $M(z)$ be an integrable function used to approximate the distributional derivative $Q'(z) = 2\delta(z)$, where $\delta(z)$ is the Dirac's delta distribution. For example, the normalized rectangular STE with half-width h, that satisfies

$$M_h^{\mathrm{rect}}(z) := \frac{1}{h}\mathbf{1}\{|z| < h\} \xrightarrow{h\to 0} 2\delta(z) \tag{2}$$

in the sense of distributions. Updating θ by standard backpropagation requires the derivative $\partial_z f(sQ(z))$. Its classical derivative is undefined at the quantization threshold and zero almost everywhere else, and therefore does not provide a useful training signal. In the sense of distributions,

$$\partial_z f(sQ(z)) = \big(f(s) - f(-s)\big)\delta(z). \tag{3}$$

This is the standard rule that a jump contributes its magnitude times a Dirac measure; see, e.g., Friedlander & Joshi (1998). Consequently, under technical conditions, [1] the exact population gradient is

$$G := \nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_x\big[\big(f(s) - f(-s)\big)\delta(g_\theta(x))\nabla_\theta g_\theta(x)\big]. \tag{4}$$

By contrast, standard QAT applies the classical chain rule at the selected quantization level and uses M in place of the quantizer distributional derivative. Its population surrogate gradient is

$$G_{\mathrm{QAT}} := \mathbb{E}_x[sf'(sq)M(g_\theta(x))\nabla_\theta g_\theta(x)]. \tag{5}$$

Thus, standard QAT weights M by the tangent $sf'(sq)$ at the selected quantization level, whereas Equation (4) is determined by the finite loss jump across the threshold.

2.2 QAT BIAS

Let $\mathcal{A}_M$ denote the error from replacing the quantizer distributional derivative $Q'(z) = 2\delta(z)$ by $M(z)$, before weighting by the downstream loss:

$$\mathcal{A}_M := \mathbb{E}_x[M(z)\nabla_\theta g_\theta(x)] - \mathbb{E}_x[2\delta(z)\nabla_\theta g_\theta(x)]. \tag{6}$$

The population bias of the standard QAT gradient then decomposes as follows.

Proposition 1 (Bias of the standard QAT activation gradient)**.** *Assume $f'(\pm s)$ exist and that $\mathbb{E}_x[|M(z)|\,|\partial_{\theta_j} g_\theta(x)|] < \infty$ for each parameter coordinate j. Then, componentwise in θ,*

$$G_{\mathrm{QAT}} - G = \underbrace{\mathbb{E}_x\left[\left(sf'(sq) - \frac{f(s)-f(-s)}{2}\right)M(z)\nabla_\theta g_\theta(x)\right]}_{B_{\mathrm{chain}}(M)} + \underbrace{\frac{f(s)-f(-s)}{2}\mathcal{A}_M}_{B_{\mathrm{STE}}(M)}. \tag{7}$$

Proof. Adding and subtracting $\frac{f(s)-f(-s)}{2}\mathbb{E}_x[M(z)\nabla_\theta g_\theta(x)]$ from $G_{\mathrm{QAT}} - G$ gives:

$$\begin{aligned} G_{\mathrm{QAT}} - G &= \mathbb{E}_x[sf'(sq)M(z)\nabla_\theta g_\theta(x)] - \frac{f(s)-f(-s)}{2}\mathbb{E}_x[2\delta(z)\nabla_\theta g_\theta(x)] \\ &= \mathbb{E}_x\left[\left(sf'(sq) - \frac{f(s)-f(-s)}{2}\right)M(z)\nabla_\theta g_\theta(x)\right] \\ &\quad + \frac{f(s)-f(-s)}{2}\mathbb{E}_x[(M(z) - 2\delta(z))\nabla_\theta g_\theta(x)]. \end{aligned} \tag{8}$$

The last expectation is $\mathcal{A}_M$, proving the decomposition. □

With M fixed, B_{chain} is the error from using the QAT coefficient $sf'(sq)$ instead of the jump coefficient $(f(s) - f(-s))/2$. With the jump coefficient fixed, B_{STE} is the error from using M instead of 2δ.

[1] We hold s and f fixed with respect to θ and assume $g_\theta(x)$ is differentiable in θ almost surely, and has regular zero crossings: $\Pr(g_\theta(x) > 0)$ is differentiable, and its derivative equals the finite boundary term $\mathbb{E}_x[\delta(g_\theta(x))\nabla_\theta g_\theta(x)]$.

2.3 CURVATURE CORRECTION

To reduce the chain-rule bias B_{chain} while keeping M unchanged, we add a curvature correction to the QAT coefficient $sf'(sq)$. Assume further that f'' is continuous on $[-s, s]$ and $f^{(3)}$ exists on $(-s, s)$. A third-order expansion about the selected value sq (detailed in Section A) gives

$$\frac{f(s) - f(-s)}{2} = sf'(sq) - s^2 q f''(sq) + \frac{2}{3}s^3 f^{(3)}(\kappa_q), \qquad \kappa_q \in (-s, s). \tag{9}$$

This motivates replacing the standard QAT coefficient $sf'(sq)$ by

$$c_{\text{corr}}(q) := sf'(sq) - s^2 q f''(sq) \tag{10}$$

and gives the corrected population gradient estimator

$$G_{\text{corr}} := \mathbb{E}_x\left[\left(sf'(sq) - s^2 q f''(sq)\right) M(g_\theta(x))\nabla_\theta g_\theta(x)\right]. \tag{11}$$

Repeating the add-and-subtract step in Equation (8) with the corrected coefficient gives

$$G_{\text{corr}} - G = \underbrace{\mathbb{E}_x\left[\left(sf'(sq) - s^2 q f''(sq) - \frac{f(s) - f(-s)}{2}\right) M(g_\theta(x))\nabla_\theta g_\theta(x)\right]}_{B^{\text{corr}}_{\text{chain}}(M)} + B_{\text{STE}}(M). \tag{12}$$

Substituting Equation (9) into Equations (7) and (12) makes the two chain-rule bias terms explicit:

$$\begin{aligned} B_{\text{chain}}(M) &= s^2\mathbb{E}_x[q f''(sq) M(z)\nabla_\theta g_\theta(x)] + B^{\text{corr}}_{\text{chain}}(M), \\ B^{\text{corr}}_{\text{chain}}(M) &= -\frac{2}{3}s^3\mathbb{E}_x\left[f^{(3)}(\kappa_q) M(z)\nabla_\theta g_\theta(x)\right]. \end{aligned} \tag{13}$$

For fixed M and bounded derivatives of f, the corrected chain-rule bias term is $O(s^3)$, compared with $O(s^2)$ for standard QAT, while both estimators retain the same approximation term $B_{\text{STE}}(M)$ from Equation (7). Note the order comparison is relevant when the quantization output scale $s < 1$, as we find to be typical for the activations studied here (see Section B.3).

2.4 PRACTICAL CURVATURE CORRECTION

Since the activation quantizer acts elementwise, the scalar curvature correction extends coordinate-wise to multidimensional activations. At layer ℓ, let $a_\ell = Q_\ell(z_\ell)$ have output levels $\pm s_\ell$, and let $f_\ell(a_\ell)$ be the downstream loss. For the theoretical correction, assume f_ℓ is twice differentiable in a_ℓ, and write $g_\ell = \nabla_{a_\ell} f_\ell(a_\ell)$ and $H_\ell = \nabla^2_{a_\ell} f_\ell(a_\ell)$. The activation-level counterpart of Equation (11) is, for each coordinate i,

$$\left(g^{\text{corr}}_{z_\ell}\right)_i = M_{\ell,i}\left(g_{\ell,i} - a_{\ell,i} H_{\ell,ii}\right), \tag{14}$$

where M_ℓ is the baseline surrogate for the quantizer derivative. Only diagonal curvature appears because crossing a threshold changes one quantized coordinate.

Computing the diagonal of the activation Hessian requires second derivatives and is impractical at modern language-model scale. We therefore make two approximations to Equation (14). First, we replace the Hessian diagonal by one inexpensive empirical-Fisher-style scalar per activation quantizer. At step t, we compute the mean squared downstream gradient and update its moving average with decay $\beta \in [0, 1)$ ($\beta = 0.9$ in our experiments):

$$\widehat{\kappa}_{\ell,t} := \text{mean}\left(g^{\odot 2}_{\ell,t}\right), \qquad \kappa_{\ell,t+1} = \beta\kappa_{\ell,t} + (1-\beta)\widehat{\kappa}_{\ell,t}. \tag{15}$$

Second, we replace the fixed offset $-a_\ell$ by the clipped quantization residual

$$r_\ell := \text{clip}(z_\ell - a_\ell, -s_\ell, s_\ell). \tag{16}$$

For the binary quantizer, $r_{\ell,i}$ approaches $-a_{\ell,i}$ as $z_{\ell,i}$ approaches its threshold from either side, and reduces to 0 where $z_\ell \approx a_\ell$. We find this approximation working better in practice, as reported in Section B.4.

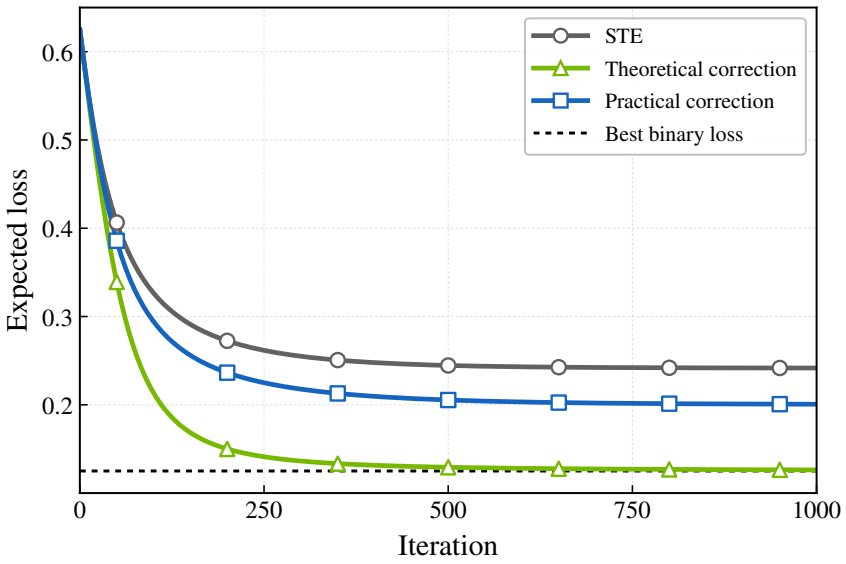


Figure 2: **Curvature correction improves optimization in a binary neuron.** We optimize the bias b in $q = \text{sign}(x + b)$, with $x \sim \mathcal{N}(0, 1)$, to minimize the loss $\mathbb{E}[\frac{1}{2}(q - 0.5)^2]$. The target is 0.5, but the output is restricted to $\{-1, +1\}$; therefore, the optimal binary prediction is always $+1$. At $q = +1$, however, the local loss slope favors decreasing the output toward 0.5. Such a change would improve a continuous prediction—for example, moving from 1 to 0.9—but is unavailable to the binary neuron. However, the only lower output available to the binary neuron is -1, whose loss is higher than that of $+1$. This mismatch motivates a curvature correction that accounts for the finite output change. The theoretical correction in Equation (11) uses the exact curvature $f''(a) = 1$ of the quadratic downstream loss. The practical correction in Equation (17) approximates this update using a clipped quantization residual and a running average of squared gradients as a nonnegative curvature proxy. The theoretical correction approaches the optimal binary loss of 0.125, while the practical correction also improves on STE. Curves use numerical population expectations, starting from $b = 0$.

Broadcasting $\kappa_{\ell,t}$ over the activation tensor and treating the residual and gradient statistic as fixed gives our practical backward rule:

$$\boxed{\widetilde{g}_{z_\ell,t} = M_{\ell,t} \odot [g_{\ell,t} + \alpha \kappa_{\ell,t} r_{\ell,t}]}. \tag{17}$$

This rule uses quantities available in the ordinary QAT forward and backward passes and stores only one moving-average scalar per quantizer. We introduce an additional hyperparameter α to compensates for the scale of the nonnegative curvature proxy. The baseline surrogate M_ℓ is unchanged.

In Figure 2, we compare, in a binary neuron simulation, ordinary QAT with STE, the exact curvature correction in Equation (11) and the practical correction in Equation (17). In this simulation, the exact-curvature rule approaches the optimal binary loss, while the practical rule improves on STE without reaching that optimum.

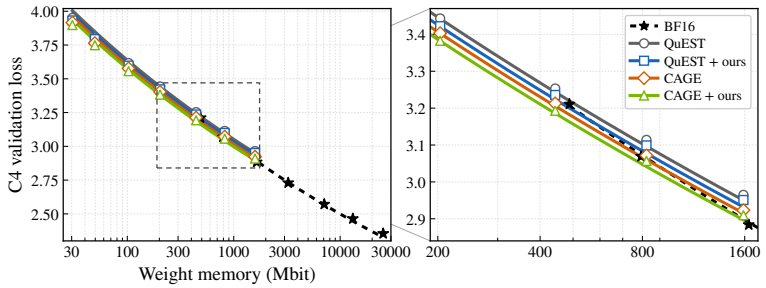
3 Experiments

3.1 Experimental setup

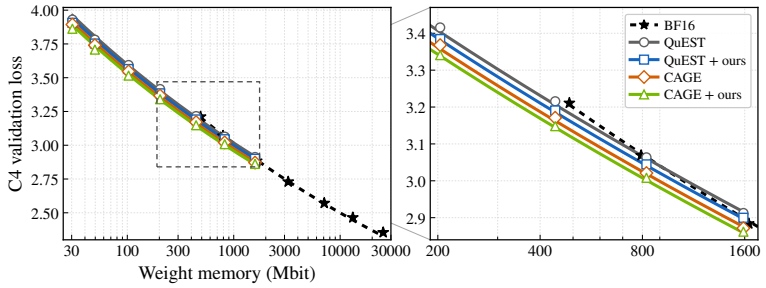
Models, data, and metric. We train decoder-only Llama-style models (Touvron et al., 2023) from scratch on the C4 dataset (Dodge et al., 2021), using the Llama 2 tokenizer and a sequence length of 512. We consider models with 30M to 1.6B non-embedding parameters and train each model for $100N$ tokens, where N is the number of non-embedding parameters. We report final perplexity and cross-entropy loss on the official C4 English validation split. We use the AdamW optimizer with cosine learning-rate scheduler with 10% warmup, weight decay 0.1, and gradient clipping at 1.0,

Table 1: **Final C4 validation perplexity.** Lower is better. The best A1W1 result at each size is bold. Each entry represents a single seed.

Transform	Method	30M	50M	100M	200M	430M	800M	1.6B
No HT	QuEST	51.95	44.95	37.20	31.28	25.87	22.51	19.39
	QuEST + ours	50.98	44.41	36.50	30.64	25.42	22.17	19.12
	CAGE	50.19	43.11	35.78	30.06	24.85	21.58	18.60
	CAGE + ours	49.31	42.47	35.04	29.45	24.36	21.26	18.35
HT	QuEST	50.92	43.87	36.37	30.44	24.92	21.40	18.40
	QuEST + ours	49.72	42.64	35.30	29.50	24.31	21.00	18.17
	CAGE	49.09	42.10	34.70	29.02	23.83	20.51	17.67
	CAGE + ours	**47.57**	**40.77**	**33.64**	**28.24**	**23.29**	**20.25**	**17.50**
BF16 reference		24.80	21.52	17.88	15.33	13.08	11.76	10.53



(a) Without HT: full range (left) and zoom (right).



(b) With HT: full range (left) and zoom (right).

Configuration	eff (m)	Gain
Without HT		
QuEST	0.0173	–
QuEST + ours	0.0193	+11%
CAGE	0.0212	–
CAGE + ours	0.0236	+11%
With HT		
QuEST	0.0212	–
QuEST + ours	0.0236	+11%
CAGE	0.0268	–
CAGE + ours	0.0295	+10%
BF16	1.0000	–

(c) Fitted parameter efficiency.

Figure 3: **Validation-loss scaling with weight memory.** In (a)–(b), the right panels enlarge the boxed regions on the left. Markers show individual runs; lines show the fitted scaling law in Equation (20). Panel (c) reports fitted parameter efficiencies, with gains relative to the matching uncorrected baselines.

following the QuEST and CAGE pretraining recipes (Panferov et al., 2025; Tabesh et al., 2025). Detailed model configurations and used hardware are reported in Section C.

Quantization baselines. We binarize weights and activations following QuEST (A1W1) (Panferov et al., 2025). For a row x, the one-bit forward quantizer used is

$$\widehat{x} = \alpha^\star \,\mathrm{RMS}(x)\,\mathrm{sign}(x), \tag{18}$$

where $\alpha^\star = \sqrt{2/\pi}$ minimizes the expected squared quantization error under a Gaussian-input assumption. Treating $\alpha^\star\,\mathrm{RMS}(x)$ as fixed in the backward pass, the gradient surrogate is

$$\widehat{\nabla_x \mathcal{L}} = \mathbf{1}\{|x| \le s\alpha^\star\,\mathrm{RMS}(x)\} \odot \nabla_{\widehat{x}}\mathcal{L}\,, \tag{19}$$

where s is a hyperparameter, selected as $s = 1.3$ in QuEST for both weights and activations. Following our analysis in Section 2, we identify that the choice of s may affect the "STE Bias", and hypothesize the optimal value may be distinct for weights and activations. Thus, we tune the s

separately for activations and weights. We obtain optimal results with $s_a = 1.1$ for activations and $s_w = 1.5$ for weights, reducing validation loss over paired model sizes by an average of 1.26%; see Section B. We also evaluate the QuEST-HT variant that applies a block-Hadamard transform before quantization and the inverse transform afterward; in this case the backward rule is applied in the transformed domain.

We additionally evaluate our method on CAGE (Tabesh et al., 2025), which augments each training step in the last $(1-\rho)$ portion of training with an auxiliary update that pulls the full-precision weights toward their quantized values, $\lambda(r_t-\rho)/(1-\rho)(\theta_{t+1}-Q(\theta_{t+1}))$, linearly increasing where $r_t = t/T$ is the ratio of the current iteration and total iterations. Following Tabesh et al. (2025), we use $\lambda = 2$ and $\rho = 0.9$, leaving the correction inactive for the first 90% of training and ramping it linearly over the final 10%.

Our correction. Similar to CAGE's protocol, we activate the correction in Equation (17) only for the final 10% of training. In our case, this design choice is related to the empirical Fisher proxy, which better approximate the Hessian near the optimum (Kunstner et al., 2019). We tune the proxy scale α on the 100M QuEST model; the sweeps are reported in Section B.4. Without HT, we use $\alpha = 3 \times 10^4$ through 430M and 2×10^4 at 800M and 1600M; with HT, we use $\alpha = 2 \times 10^4$ through 200M and 3×10^4 from 430M onward.

3.2 Results

Table 1 reports the final validation perplexity for all eight A1W1 configurations. Our correction improves both QuEST and CAGE at every measured model size, with and without HT. Without HT, it reduces perplexity by 1.2%–2.1% over QuEST and by 1.4%–2.1% over CAGE. With HT, the corresponding reductions are 1.3%–3.1% over QuEST and 1.0%–3.2% over CAGE.

To evaluate the loss scaling law of our method, we follow QuEST and CAGE (Panferov et al., 2025; Tabesh et al., 2025) and model final validation cross-entropy as

$$L(N, D, m) = \frac{A}{\big(N \operatorname{eff}(m)\big)^{\gamma}} + \frac{B}{D^{\beta}} + E, \tag{20}$$

where D is the number of training tokens and m indexes the training configuration. We fix (A, γ, B, β, E) to the released QuEST values and fit only $\operatorname{eff}(m)$, using the exact parameter and token counts and the paper-specified Huber objective with $\delta = 10^{-3}$ on log cross-entropy. The results of the fits are shown in Figure 3, and the results $\operatorname{eff}(m)$ are shown in Figure 3c. The fit process details are reported in Section B.2.

The fitted efficiencies reinforce the pointwise results. Our correction increases effective parameter count by 11.2% over QuEST and 11.1% over CAGE without HT, and by 11.5% and 10.3%, respectively, with HT. The correction therefore provides a similar relative capacity gain across both base recipes and transform domains.

4 Related Work

Quantization is a key area of research in the effort to compress neural networks for more efficient inference. Quantization-aware training (QAT) incorporates quantization effects during model training or fine-tuning, enabling the network to adapt and maintain accuracy under low-precision constraints. In QAT, the model evaluates a hard-quantized model in the forward pass while retaining latent parameters in high precision and applies a straight-through estimator (STE) or a clipped variant in the backward pass (Bengio et al., 2013; Hubara et al., 2016). Early methods improved the forward quantizer by learning clipping ranges or step sizes (Choi et al., 2018; Esser et al., 2020). Recent work has extended QAT to language models at extremely low precision. QuEST combines MSE-optimal distribution fitting, Hadamard preprocessing, and an error-based trust estimator, enabling stable pretraining down to W1A1 (Panferov et al., 2025). ParetoQ studies low-bit weight-only QAT, primarily through fine-tuning, and finds binary models less favorable than ternary, 2-bit, and 3-bit models in its size–accuracy comparisons (Liu et al., 2025). More recently, BBQ builds on QuEST's block-Hadamard preprocessing and uses a Gaussian-CDF transform to distribute values more evenly

across uniform quantization bins, mapping the resulting codes to hardware-supported low-precision formats (Yang & Aamodt, 2026).

Several studies examine the gradient propagated through quantized activations. Yin et al. (2019) analyze when an STE provides a descent direction in a two-layer model. EWGS (Lee et al., 2021) and PEGE (Zhao, 2025) use quantization residuals to adjust gradients: EWGS calibrates gradient scaling using Hessian estimates, whereas PEGE schedules both forward quantization and the correction coefficient. Both were evaluated on convolutional networks for image classification. SURGE uses a full-precision auxiliary branch to compensate gradients in binary networks, including BERT fine-tuning on GLUE (Huang et al., 2026). Our method instead addresses the chain-rule mismatch between the STE coefficient and finite loss jump revealed by our distributional-derivative analysis, using a practical curvature correction evaluated in A1W1 autoregressive LLM pretraining, a setting not studied by these methods.

Other low-bit LLM QAT methods focus on the updates of the latent weights. LOTION (Kwun et al., 2025) smooths the loss under unbiased randomized rounding of weights, practically adding a curvature-aware regularizer; its practical approximation adds a curvature-aware regularizer, and its LLM experiments use weight-only quantization. CAGE (Tabesh et al., 2025) uses the residual $\theta - Q(\theta)$ in a correction applied after the optimizer step that pulls latent weights toward the quantization grid, improving QuEST in W2A2–W4A4 training. Our experiments show that CAGE also improves QuEST in the binary case and that our activation correction provides further gains when combined with CAGE. WinQ (Li et al., 2026) periodically interpolates latent and quantized weights and injects noise to accelerate escape from flat saddle regions.

FOGZO and JacQuant target parameter-gradient estimation in low-bit QAT. FOGZO mixes an STE direction with zeroth-order information (Yang & Aamodt, 2025). JacQuant learns a diagonal or groupwise surrogate Jacobian for weight gradients in ultra-low-bit LLM QAT; its reported experiments use 8- or 16-bit activations (Yi et al., 2026). Neither addresses the activation-threshold chain-rule mismatch studied here.

5 Discussion

5.1 Conclusions

In this work, we examined activation gradients in the training of binarized neural networks. In Proposition 1 we showed that their bias separates into an error from the surrogate quantizer derivative and a chain-rule error from using the loss slope at a quantization level in place of the finite loss jump between levels. We address the chain-rule error by correcting the coefficient propagated through the activation quantizer using loss curvature. Under the assumptions of our analysis in Section 2, our curvature correction reduces the order of the chain-rule term for a fixed quantizer-derivative surrogate. We make this correction practical with a bounded quantization residual and an empirical-Fisher-style curvature proxy.

In the binary-neuron simulation presented in Figure 2, the exact-curvature correction approaches the optimal binary loss, while the practical rule improves on standard STE training. In one-bit Llama-style models from 30M to 1.6B parameters, applying the practical rule during the final 10% of training reduces final C4 validation perplexity by 1.0–3.2% relative to the corresponding QuEST or CAGE baseline at every evaluated size, with and without HT (see Table 1). Both CAGE and our method build on QuEST and target different parts of training: CAGE pulls latent weights toward their quantized values (Tabesh et al., 2025), while our method addresses the activation-gradient bias. Notably, their gains are approximately additive in perplexity points: across the evaluated sizes, the combined reduction relative to QuEST is close to the sum of the reductions achieved by CAGE and our correction separately.

5.2 Limitations and Future Work

Our analysis identifies two sources of activation-gradient bias—the "STE bias" and the "chain-rule bias"—yet our method addresses only the latter. The STE bias arises from approximating the quantizer derivative with a surrogate and depends on the choice of that surrogate. Future work could characterize this bias and develop surrogates that reduce it. Because the two bias components add,

they may partly cancel when they point in opposing directions, so reducing the chain-rule term alone theoretically may not always reduce total bias. The consistent empirical gains show that this possibility does not prevent improvements in the evaluated models, although our experiments do not measure the bias components directly.

Related work on fully quantized LLM training highlights the importance of unbiased updates when backward gradients are numerically quantized (Chmiel et al., 2025; Panferov et al., 2026). However, a correction intended to reduce activation-gradient bias could also increase the variance of stochastic gradients and hinder optimization. Future work can build on our derivations and measure this tradeoff for practical activation-gradient corrections.

AI USE STATEMENT

In this work, we used generative AI tools, including Codex and Claude Code, to assist with drafting and editing the manuscript, including textual content and mathematical derivations; implementing training and figure-generation scripts; and monitoring the execution of training scripts. We have not used generative AI tools to generate synthetic datasets; develop the proposed method, theoretical models, or conceptual frameworks; formulate mathematical claims; propose or refine hypotheses; derive the theoretical proofs or provide critical ingredients for them; design or provide feedback on research methodology or experiments; translate material; clean or reformat datasets; support qualitative or thematic data analysis; or interpret results. We have reviewed all AI-assisted work. We checked every output of these tools, including manuscript text and code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation, August 2013. URL http://arxiv.org/abs/1308.3432. arXiv:1308.3432 [cs].

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. FP4 All the Way: Fully Quantized Training of LLMs, August 2025. URL http://arxiv.org/abs/2505.19115. arXiv:2505.19115 [cs.LG].

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I.-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks, July 2018. URL http://arxiv.org/abs/1805.06085. arXiv:1805.06085 [cs.CV].

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus, September 2021. URL http://arxiv.org/abs/2104.08758. arXiv:2104.08758 [cs.CL].

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned Step Size Quantization, May 2020. URL http://arxiv.org/abs/1902.08153. arXiv:1902.08153 [cs.LG].

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, March 2023. URL http://arxiv.org/abs/2210.17323. arXiv:2210.17323 [cs.LG].

F. G. Friedlander and M. Joshi. *Introduction to the Theory of Distributions*. Cambridge University Press, 1998. ISBN 978-0-521-64971-1.

Haoyu Huang, Boyu Liu, Linlin Yang, Yanjing Li, Yuguang Yang, Xuhui Liu, Canyu Chen, Zhongqian Fu, and Baochang Zhang. SURGE: Surrogate Gradient Adaptation in Binary Neural Networks, May 2026. URL http://arxiv.org/abs/2605.10989. arXiv:2605.10989 [cs.LG].

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.neurips.cc/paper_files/paper/2019/hash/46a558d97954d0692411c861cf78ef79-Abstract.html.

Mujin Kwun, Depen Morwani, Chloe Huangyuan Su, Stephanie Gil, Nikhil Anand, and Sham Kakade. LOTION: Smoothing the Optimization Landscape for Quantized Training, October 2025. URL http://arxiv.org/abs/2510.08757. arXiv:2510.08757 [cs.LG].

Junghyup Lee, Dohyung Kim, and Bumsub Ham. Network Quantization with Element-wise Gradient Scaling, April 2021. URL http://arxiv.org/abs/2104.00903. arXiv:2104.00903 [cs.CV].

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, May 2026. URL http://arxiv.org/abs/2605.17471. arXiv:2605.17471 [cs.LG].

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration, April 2026. URL http://arxiv.org/abs/2306.00978. arXiv:2306.00978 [cs.CL].

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, October 2025. URL http://arxiv.org/abs/2502.02631. arXiv:2502.02631 [cs.LG].

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L. Castro, Mahdi Nikdan, and Dan Alistarh. QuEST: Stable Training of LLMs with 1-Bit Weights and Activations, June 2025. URL http://arxiv.org/abs/2502.05003. arXiv:2502.05003 [cs.LG].

Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet II: Accurate LLM Pre-Training in NVFP4 by Improved Unbiased Gradient Estimation, June 2026. URL http://arxiv.org/abs/2601.22813. arXiv:2601.22813 [cs.LG].

Soroush Tabesh, Mher Safaryan, Andrei Panferov, Alexandra Volkova, and Dan Alistarh. CAGE: Curvature-Aware Gradient Estimation For Accurate Quantization-Aware Training, November 2025. URL http://arxiv.org/abs/2510.18784. arXiv:2510.18784 [cs.LG].

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288. arXiv:2307.09288 [cs.CL].

Ningfeng Yang and Tor M. Aamodt. Improving the Straight-Through Estimator with Zeroth-Order Information, October 2025. URL http://arxiv.org/abs/2510.23926. arXiv:2510.23926 [cs.LG].

Ningfeng Yang and Tor M. Aamodt. Boosting Entropy with Bell Box Quantization, March 2026. URL http://arxiv.org/abs/2603.01599. arXiv:2603.01599 [cs.LG].

Kai Yi, Vignesh Vivekraja, Harshit Khaitan, and Steven Li. JacQuant: STE-Free Quantization-Aware Training via Learned Jacobian Surrogates, May 2026. URL https://arxiv.org/abs/2605.25469v1.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets, September 2019. URL http://arxiv.org/abs/1903.05662. arXiv:1903.05662 [cs.LG].

Kaiqi Zhao. Progressive Element-wise Gradient Estimation for Neural Network Quantization, August 2025. URL http://arxiv.org/abs/2509.00097. arXiv:2509.00097 [cs.LG].

A Curvature-Correction Derivation

We give the Taylor steps behind Equation (9) and the corrected bias comparison in Equation (13).

Fix $q \in \{-1, +1\}$. The displacement from sq to the other quantization level $-sq$ is $-2sq$. Taylor's theorem with a third-order Lagrange remainder gives some $\kappa_q \in (-s, s)$ for which

$$\begin{aligned} f(-sq) &= f(sq) - 2sqf'(sq) + \frac{(-2sq)^2}{2} f''(sq) + \frac{(-2sq)^3}{6} f^{(3)}(\kappa_q) \\ &= f(sq) - 2sqf'(sq) + 2s^2 f''(sq) - \frac{4}{3} s^3 q f^{(3)}(\kappa_q), \end{aligned} \tag{21}$$

where the last line uses $q^2 = 1$. For either sign, the endpoint difference satisfies $f(sq) - f(-sq) = q\big(f(s) - f(-s)\big)$. Subtracting Equation (21) from $f(sq)$ therefore gives

$$q\big(f(s) - f(-s)\big) = 2sqf'(sq) - 2s^2 f''(sq) + \frac{4}{3} s^3 q f^{(3)}(\kappa_q). \tag{22}$$

Multiplying by $q/2$ and again using $q^2 = 1$ gives

$$\frac{f(s) - f(-s)}{2} = sf'(sq) - s^2 q f''(sq) + \frac{2}{3} s^3 f^{(3)}(\kappa_q),$$

which is Equation (9). By Equation (10), the first two terms on its right-hand side equal $c_{\mathrm{corr}}(q)$. Subtracting the resulting jump coefficient gives the exact error

$$\begin{aligned} c_{\mathrm{corr}}(q) - \frac{f(s) - f(-s)}{2} &= c_{\mathrm{corr}}(q) - \left(c_{\mathrm{corr}}(q) + \frac{2}{3} s^3 f^{(3)}(\kappa_q) \right) \\ &= -\frac{2}{3} s^3 f^{(3)}(\kappa_q). \end{aligned} \tag{23}$$

B Additional Results

B.1 Clip-scale sweep

The clipping multiplier s in Equation (19) controls the support of the STE. QuEST (Panferov et al., 2025) uses one shared multiplier for weights and activations, with its A1W1 sweep selecting values near 1.25–1.30. We use $s_a = s_w = 1.3$ as the shared-scale QuEST baseline and test whether weights and activations benefit from separate multipliers.

On the 30M model without HT, we sweep

$$s_a \in \{0.5, 0.8, 1.0, 1.1, 1.2, 1.3, 1.5\}, \qquad s_w \in \{0.5, 0.8, 1.0, 1.1, 1.2, 1.3, 1.5, 1.6, 1.7, 1.8\}.$$

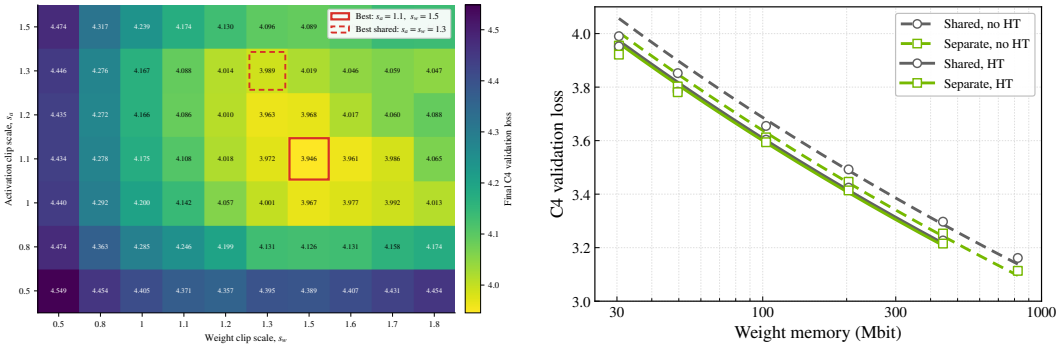
Each cell is a single-seed run with the same data, model, optimizer, and schedule. Figure 4a shows that the best observed setting is $s_a = 1.1$ and $s_w = 1.5$, with final validation loss 3.9456. The best shared setting on the diagonal is $s_a = s_w = 1.3$, with loss 3.9886. A separate HT-domain sweep selects the same pair.

We then compare the shared and separate settings across model scales, without the curvature correction or CAGE. Averaged over the paired 30M–800M runs, separate scales reduce validation loss by 0.0447 (1.26%) without HT. Over the common 50M–430M scale-up runs with HT, the average reduction is 0.0082 (0.24%). In Figure 4b, markers denote the measured runs and smooth lines use the scaling law in Equation (20), with the shared QuEST coefficients fixed and one efficiency fitted per curve. The no-HT curves contain results through 800M, while the HT curves contain results through 430M.

B.2 Fitted parameter-efficiency details

For the parameter efficiencies reported in Figure 3c, we use the model in Equation (20) and fix its five shared coefficients to the released QuEST fit:

$$(A, \gamma, B, \beta, E) = (8.13403,\ 0.0627265,\ 86.4119,\ 0.241288,\ 0.00142329).$$



(a) QuEST 30M A1W1 clip-scale grid.

(b) QuEST A1W1 scaling-law fits.

Figure 4: **Shared versus separate activation and weight clip scales. (a)** Final C4 validation loss as the activation and weight multipliers are varied independently. The solid outline marks the best cell in the full grid, and the dashed outline marks the best cell constrained to one shared multiplier. **(b)** The shared QuEST-scale baseline $s_a = s_w = 1.3$ is compared with the separately tuned setting $s_a = 1.1$, $s_w = 1.5$, with and without HT. Lines show the fitted scaling law.

We fit only one positive $\mathrm{eff}(m)$ independently for each of the eight A1W1 configurations. Writing $\eta_m = \log \mathrm{eff}(m)$, the fit minimizes

$$\widehat{\eta}_m = \arg\min_{\eta \in \mathbb{R}} \sum_i \mathcal{H}_{10^{-3}} \left(\log L_i - \log \widehat{L}(N_i, D_i; e^{\eta}) \right), \tag{24}$$

where $\mathcal{H}_\delta$ is the Huber loss. We use the exact non-embedding parameter and token counts for all seven model sizes.

B.3 Quantizer scale measurments

Our curvature correction reduces the $O(s^2)$ term in the "chain-rule" bias to $O(s^3)$ (Equation (13)), which is relevant when the quantizer output scale $s < 1$. We provide empirical evidence that this is typical for the activations in our models. Following QuEST (Panferov et al., 2025), each activation row x in our model has output levels $\pm s_x$, with $s_x \approx \sqrt{2/\pi}\,\mathrm{RMS}(x)$. We measure s_x directly during training and report them in Figure 5. Here $\bar{s}_x$ denotes the mean of s_x over activation rows in a sampled training microbatch at a given layer. At each plotted training point, the boxes summarize the distribution of $\bar{s}_x$ across activation quantization layers.

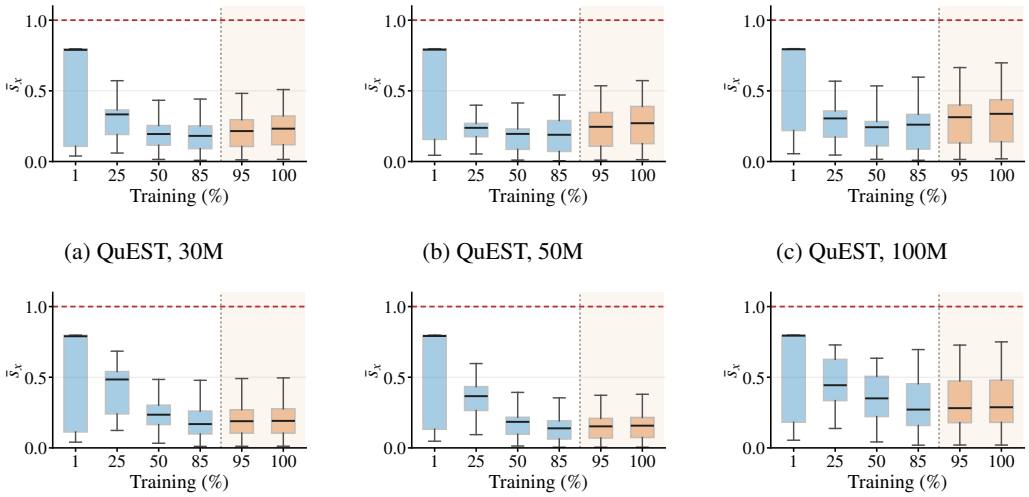
B.4 Practical curvature correction ablation study

We run an ablation study of the practical design choices of the curvature correction proposed in Section 2.4. We compare the clipped quantization residual r_ℓ with the fixed offset $-a_\ell$, and the empirical Fisher factor κ_ℓ with a fixed constant. Combining the two, we evaluate four configurations and run a sweep over α on each configuration. We repeat the sweep for 30M, 50M, and 100M A1W1 QuEST. For each size, all configurations continue from the same checkpoint at 90% of training. We report in Figure 6 the change in validation loss compared to the corresponding baseline. The results shows that the Fisher–residual configuration reaches the lowest final validation loss at all three model sizes.

C Training configurations and hardware

The BF16 references and all A1W1 methods use the same model architecture, training data, and optimizer settings at each size, following QuEST recipe Panferov et al. (2025). Table 2 lists settings shared across sizes. Table 3 gives the architecture and size-dependent settings.

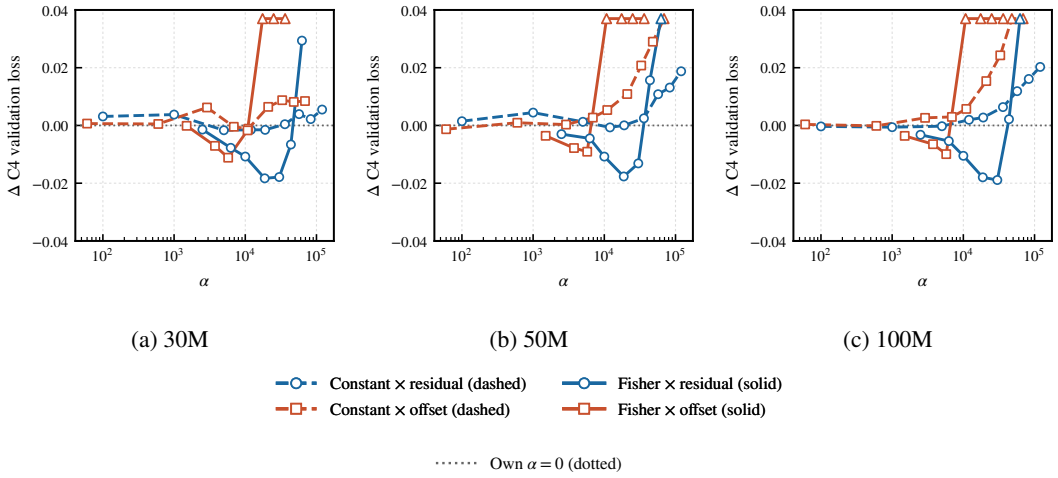
A1W1 methods. Weights and activations use centered, row-wise, one-bit MSE quantization with STE clipping multipliers $s_w = 1.5$ and $s_a = 1.1$. HT variants apply a normalized 128-dimensional



(a) QuEST, 30M (b) QuEST, 50M (c) QuEST, 100M

(d) QuEST+HT, 30M (e) QuEST+HT, 50M (f) QuEST+HT, 100M

Figure 5: **Activation quantizer output scales during training.** Boxes span the 25th–75th percentiles, and the horizontal line marks the median (50th percentile). Blue boxes precede curvature correction and orange boxes follow its 90% start.



(a) 30M (b) 50M (c) 100M

Figure 6: **Practical curvature correction ablation.** Change in final C4 validation loss relative to each configuration's baseline (lower is better). Triangles mark losses above the plotted range.

block-Hadamard transform. CAGE uses $\lambda = 2$, starts at training fraction $\rho = 0.9$, and ramps linearly over the remaining steps. Our activation-gradient correction also starts at 90% of training; its empirical-Fisher scalar uses EMA decay $\beta = 0.9$ and the size-dependent α shown in Table 3. The final 10% of corrected runs continue from the corresponding baseline checkpoint at the same training step.

Hardware. We used one NVIDIA H100 GPU for the 30M and 50M models, one node with eight H100 GPUs for the 100M–800M models, and four nodes with eight H100 GPUs each for the 1.6B models.

Table 2: Shared training hyperparameters for the scaling experiments.

Setting	Value
Data and tokenizer	C4 training split, tokenized with the Llama 2 tokenizer; C4 validation for final loss and perplexity
Sequence length and vocabulary	512 tokens; 32,000 tokens
Effective global batch	512 sequences = 262,144 tokens per optimizer step
Optimizer	AdamW; $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$
Weight decay	0.1 on linear weights; none on normalization, embedding, or output-projection weights
Gradient clipping	Global norm at 1.0
Learning-rate schedule	Cosine decay after warmup; initial rate $\eta_{\max}/100$, final rate $\eta_{\max}/10$
Architecture details	SwiGLU feed-forward layers; 128-dimensional attention heads; RMSNorm with $\epsilon = 10^{-5}$; rotary embeddings with base 10^4; no dropout or linear bias

Table 3: Architecture and size-dependent training configurations. d is the model width, H the number of attention heads, and d_{ff} the SwiGLU hidden width. T is the number of optimizer steps, T_{w} the warmup steps, and D the realized number of training tokens in billions. The two α columns apply only to our corrected A1W1 runs.

Size	N (M)	L	d	H	d_{ff}
30M	30.483	6	640	5	1792
50M	49.557	7	768	6	2048
100M	102.778	8	1024	8	2816
200M	203.188	10	1280	10	3584
430M	443.068	13	1664	13	4608
800M	822.151	16	2048	16	5632
1.6B	1586.076	20	2560	20	6912

Size	$\eta_{\max}$	T	T_{w}	D (B)	α (no HT)	α (HT)
30M	1.2×10^{-3}	11,444	1,144	3	3×10^4	2×10^4
50M	1.2×10^{-3}	19,073	1,907	5	3×10^4	2×10^4
100M	6.0×10^{-4}	38,146	3,814	10	3×10^4	2×10^4
200M	3.0×10^{-4}	76,293	7,629	20	3×10^4	2×10^4
430M	1.5×10^{-4}	164,030	16,403	43	3×10^4	3×10^4
800M	7.5×10^{-5}	305,170	30,517	80	2×10^4	3×10^4
1.6B	3.75×10^{-5}	610,350	61,035	160	2×10^4	3×10^4