OSFP4: JOINT OPTIMIZATION OF DIAGONAL SMOOTHING AND BLOCK SCALES FOR NVFP4 QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

NVFP4 is an attractive datatype for large language model (LLM) inference, offering compact storage and native tensor-core acceleration. However, preserving accuracy using NVFP4 requires careful quantization. In this work we develop a novel quantization scheme called Optimized Smoothing and Scaling for NVFP4 (OSFP4). For each linear projection it uses a diagonal smoothing matrix whose entries are optimized to minimize the squared matrix-product quantization error under NVFP4, taking into account the rounding procedure that is used (either round-to-nearest, or GPTQ-style successive interference cancellation). This requires performing joint optimization on the smoothing entries as well as the block scales, which is facilitated by analyzing a multiplicative-dither FP4 quantizer instead of the fixed deterministic one. Experiments show that OSFP4 achieves the highest average accuracy among the evaluated competitors in the corresponding quantization settings, while retaining approximately 94-97% of vendor NVFP4 prefill throughput on the measured workloads.

1 INTRODUCTION

Low-bit data formats are gaining much popularity for large language models (LLMs) inference, as they require less memory for storing and running the model, decrease the traffic from and to memory, and allow to execute matrix multiplication (MatMul) faster through optimized hardware. The FP4 datatype stands out as a prominent representative of such formats which is heavily used in models running on NVIDIA's Blackwell series and its subsequent generations. Unfortunately, using FP4 instead of higher resolution formats such as FP8 often comes with the cost of reduced accuracy. The goal of this paper is to develop a quantization scheme for NVFP4 that jointly optimizes a *diagonal smoothing transformation, as well as the weight block scales* in order to maintain maximal accuracy.

At first thought, it is not at all clear that diagonal smoothing is useful for FP4 quantization. Traditionally, the main differentiator between INT constellations and FP constellations was the dynamic range. An INT-R constellation has fixed spacing between constellation points, and its dynamic range is therefore proportional to 2^R. Absmax scaling is often used to make sure the input fits within the dynamic range, and the corresponding per-entry mean-squared error (MSE) in representing $x \in \mathbb{R}^n$ in INT-R is consequently proportional to $\|x\|_\infty^2 \cdot 2^{-2R}$ Ordentlich & Polyanskiy (2026a). For this reason, one of the key aspects in INT-based quantization techniques is controlling the infinity norm of the quantizer's input. Diagonal scaling can help balance the infinity norms of vectors participating in MatMul, and indeed one of the simplest and most efficient techniques for INT-quantized MatMul is SmoothQuant Xiao et al. (2023).

For FP constellations with E exponent bits, the dynamic range is proportional to 2^{2^E}, and can be made "essentially unbounded" with as few as $E = 4$ exponent bits, as in the standard E4M3 FP8 datatype. The MSE in representing the ith entry x_i of $x \in \mathbb{R}^n$ is therefore proportional to $|x_i|^2 \cdot 2^{-2M}$, where M is the number of mantissa bits, provided that x_i is inside the constellation's dynamic range Ordentlich & Polyanskiy (2026a). For this reason, quantization to FP is typically considered simpler, and requires less pre and post processing than quantization to INT constellations. In particular, whenever E is large enough so that dynamic range is not a limiting factor, SmoothQuant-style diagonal scaling is quite useless, as $\mathbb{E}[(Q_{\mathrm{FP}}(X_i \cdot \alpha)Q_{\mathrm{FP}}(Y_i/\alpha) - X_iY_i)^2]$ is

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.

proportional to $\mathbb{E}[|X_i|^2|Y_i|^2]2^{-2M}$ for all $\alpha > 0$ provided that the distribution of (X_i, Y_i) is "sufficiently smooth" and $X_i \cdot \alpha$ and Y_i/α are within the quantizer's dynamic range with high probability.

The picture is different for the FP4 E2M1 format since with $E = 2$ the dynamic range is too small to be ignored. To combat the effect of limited dynamic range, FP4 is usually applied with microscaling as in MXFP4 or NVFP4, where a group of g entries are also given a common scale γ. In MXFP4 $g = 32$ and γ is represented in E8M0 format, whereas in NVFP4 $g = 16$ and γ is represented in E4M3 format. Here we focus on NVFP4, which is becoming a common choice for models running on NVIDIA hardware, though some of the techniques we develop may also be useful for MXFP4 as well.

The scale γ can be used to ensure that the input to the FP4 quantizer never exceeds 6, which is the largest value in the FP4 constellation. In particular, for $x \in \mathbb{R}^{16}$ the default absmax-to-6 rule sets $\gamma \approx \|x\|_\infty/6$ such that $\|\gamma^{-1}x\|_\infty \approx 6$. This rule takes care of the upper limit of the dynamic range: it ensures that no entry of $\gamma^{-1}x$ is large enough to saturate the quantizer. However, with this rule all entries with magnitude 24 times smaller than $\|x\|_\infty$ will be quantized to 0, so that the total distortion can be quite large when x is largely unbalanced. Furthermore, even for balanced vectors, there are multiple choices of γ for which all entries of $\gamma^{-1}x$ are within the dynamic range of the FP4 constellation, and careful optimization of γ may yield significant reduction of MSE over the absmax-to-6 rule.

This work develops an NVFP4 quantization method that utilizes a diagonal matrix $A \in \mathbb{R}^{n\times n}$ for "smoothing" the weight matrix $W \in \mathbb{R}^{m\times n}$ to WA^{-1} and the activation matrix $X \in \mathbb{R}^{n\times k}$ to AX, as in SmoothQuant. The clear advantage of diagonal A with respect to more general structures for A, e.g. block-rotation Egiazarian et al. (2026), PCA directions aligning Chen et al. (2025), tensor of dense small matrices Sun et al. (2025), is that diagonal pre-processing is cheap and can easily be done in runtime for the activations. In fact, in some layers it can be fused into the LayerNorm or RMSnorm.

In contrast to SmoothQuant where A is used for balancing infinity norms, the role of the matrix A here is more intricate. It should balance the distribution of all entries within the same groups (16 consecutive entries) in the rows of W and in the columns of X, so that the corresponding E4M3 scales can then be used to minimize the MSE. Developing a mathematical formulation that accurately captures this objective is one of the main challenges addressed in this work. This is achieved by analyzing a randomized multiplicatively dithered FP4 quantizer instead of the deterministic one, which is actually used by our scheme. The distortion incurred by the randomized quantizer admits a relatively simple form using the function $\phi(\cdot)$ introduced below (See Figure 1), and is closely related to the distortion incurred by the deterministic quantizer (Lemma 1). This in turn enables to jointly optimize the smoothing weight matrix and the block scales.

Using the fixed weights and calibration activations, we optimize the diagonal entries of A and the block scales offline in two stages. In the first stage we jointly optimize the diagonal entries of A and the block scales with respect to a dithered FP4 quantizer, whose corresponding MSE loss is smooth and amenable to optimization. Crucially, the loss takes into account whether round-to-nearest (RTN) or successive-interference-cancellation (SIC) will be used for the actual weight quantization. The resulting A is retained, while the weight scales are finely optimized over the E4M3 grid in the second stage. Activation scales must instead be determined in real time; our default is absmax-to-6.

Our main contributions are:

- We develop an expression that approximates the expected squared Frobenius norm of the matrix-product error under randomized FP4 quantization for diagonally smoothed NVFP4 quantization, as a function of the smoothing parameters and the group scales. This expression takes into account the rounding procedure (RTN/SIC) for W, and is amenable to efficient joint optimization of all participating parameters.
- Relying on this approximation, we develop a quantization scheme called Optimized Smoothing and Scaling for NVFP4 (OSFP4) quantization. This scheme uses the approximation to perform joint continuous optimization of diagonal smoothing and block scales, followed by discrete E4M3 weight-scale selection. The optimization criterion for the discrete weight-scale depends on the calibration second-moments as well as on the rounding method (RTN/SIC). The activation group scales are set using absmax-to-6.

- Experiments show that OSFP4 achieves the highest average accuracy among the evaluated competitors in the corresponding quantization settings, while retaining approximately 94-97% of vendor NVFP4 prefill throughput on the measured workloads.

- We provide a complete code package for quantizing a model to a vLLM checkpoint using OSFP4. The package is available in an OSFP4 code repository.

1.1 Related Work

Calibration and error compensation. Post-training quantization uses calibration data to preserve layer outputs without retraining the original model. GPTQ (Frantar et al., 2023) uses second-order information to compensate quantization errors through updates to unquantized weights. This method has demonstrated empirical benefits for various quantization codebooks, including for FP and in particular NVFP4 constellations Egiazarian et al. (2026). Diagonal scaling has proved useful for INT constellations: SmoothQuant (Xiao et al., 2023) balances the quantization difficulty of weights and activations by redistributing their magnitudes, while AWQ (Lin et al., 2024) uses activation statistics to scale and protect important weight channels. OmniQuant (Shao et al., 2024) learns such scalings as part of a broader family of equivalent transformations, jointly with clipping parameters, through reconstruction objectives. These methods establish the value of calibration, scaling, and feedback. WaterSIC (Lifar et al., 2026) combines all three and finds a diagonal smoothing matrix which, when combined with GPTQ/SIC, obtains near-information-theoretic optimal weight-only entropy coded integer quantization. Another work addressing optimal diagonal smoothing for weights and activations INT quantization using GPTQ was submitted to this conference (Anonymous, 2027). It turns out that NVFP4 requires a markedly different criterion for choosing the diagonal smoothing matrix. OSFP4 jointly adapts smoothing and block scales to the NVFP4 grid and the chosen rounding procedure.

Transforms for low-bit inference. Prior work demonstrated the value of randomized rotations for low-bit quantization (Chee et al., 2023; Ashkboos et al., 2024; Liu et al., 2025). For NVFP4, however, full-dimensional rotations were found less beneficial and can even hurt RTN accuracy, whereas MR-GPTQ (Egiazarian et al., 2026) benefits from rotations within quantization groups together with format-specific GPTQ optimizations. OSFP4 avoids rotations and instead uses cheaper diagonal transformations, jointly optimized (offline) with the block scales. Despite this simpler transformation, our W-SIC/X-RTN configuration achieves higher average W4A4 accuracy than MR-GPTQ under the common evaluation protocol in Table 1.

NVFP4 scale selection. Recent work shows that absmax scaling leaves substantial room for improvement. Four Over Six (4/6) (Cook et al., 2026) adaptively selects between absmax-to-4 and absmax-to-6 scaling for each block. ScaleSearch (Gupta et al., 2026) searches block scales to reduce reconstruction error, and ScaleSweep (Lin & Wan, 2026) derives bounded search ranges for MSE and weighted MSE objectives. SOAR (Bao et al., 2026) combines joint global/block-scale optimization with decoupled encoding and decoding scales. H-Scale (Yu et al., 2026) uses calibration activation second moments to select E4M3 block scales by minimizing a weighted-mean-squared weight reconstruction error. OSFP4 combines accurate scale selection with diagonal smoothing, jointly optimizing both while accounting for the applied rounding procedure (RTN/SIC). Online activation-scale search is complementary to this procedure; our default uses absmax activation scaling.

Quantization-error models. Our analysis builds on high-rate models of floating-point error and covariance-aware quantized matrix multiplication (Ordentlich & Polyanskiy, 2026a;b). We use multiplicative randomization to obtain a smooth FP4 error function that retains finite-range effects, and derive distinct objectives for RTN and SIC, with or without activation quantization. This makes the intended feedback procedure part of the smoothing optimization and connects the randomized loss to the subsequent deterministic scale search.

2 Loss Functions

2.1 Smoothed FP4 quantization error

Denote by

$$\mathcal{C}_{\mathrm{FP4}} = \left\{0, \pm\frac{1}{2}, \pm 1, \pm\frac{3}{2}, \pm 2, \pm 3, \pm 4, \pm 6\right\} \tag{1}$$

the points in the FP4 E2M1 constellation, and for $x \in \mathbb{R}$ let

$$Q_{\mathrm{FP4}}(x) = \operatorname*{argmin}_{\hat{x}\in\mathcal{C}_{\mathrm{FP4}}} |x - \hat{x}| \tag{2}$$

be the deterministic nearest-neighbor quantizer of x to FP4. Inspired by the *dithered absmax FP quantizer* from Ordentlich & Polyanskiy (2026a) we consider the following randomized FP4 quantizer

$$\tilde{Q}_{\mathrm{FP4}}(x) = 2^U Q_{\mathrm{FP4}}\left(2^{-U}x\right), \tag{3}$$

where $U \sim \mathrm{Uniform}([0,1))$ is statistically independent of x, and its corresponding normalized MSE function is

$$\phi(x) = \frac{1}{x^2}\mathbb{E}\left(x - \tilde{Q}_{\mathrm{FP4}}(x)\right)^2. \tag{4}$$

We set $\phi(0) = 1$ by continuous extension. The following lemma provides two key properties of $\tilde{Q}_{\mathrm{FP4}}(x)$ and $\phi(x)$.

Lemma 1 *1. For any $x \in \mathbb{R}^n$ there exists a fixed $\beta \in [1,2)$ such that*

$$\|x - \beta \cdot Q_{\mathrm{FP4}}(\beta^{-1}x)\|^2 \le \sum_{i=1}^{n} x_i^2 \phi(x_i). \tag{5}$$

2. $\phi(x)$ is an even function. Its minimum is $\phi_{\min} = \mathbb{E}[(1 - 2^{-(U+1)}\mathrm{round}(2^{U+1}))^2] \approx 0.010723$ and it attains this minimum on the entire interval $\mathcal{I}_{\mathrm{FP4}} = [7/4, 7]$.

A proof is given in Appendix A. Figure 1 plots $\phi(x)$ as well as the non-smoothed $\phi_{\mathrm{exact}}(x) = \frac{1}{x^2}\mathbb{E}\left(x - Q_{\mathrm{FP4}}(x)\right)^2$. The plateau in $\mathcal{I}_{\mathrm{FP4}}$ is visibly clear.

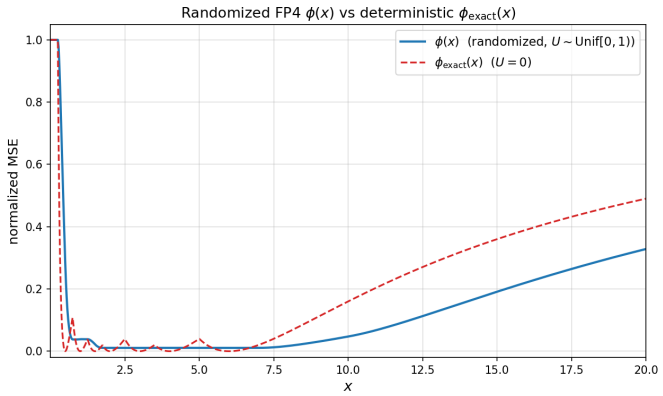


Figure 1: Normalized mean-squared error of randomized FP4 quantization, $\phi(x) = \frac{\mathbb{E}[(x-\tilde{Q}_{\mathrm{FP4}}(x))^2]}{x^2}$, and its deterministic counterpart, $\phi_{\mathrm{exact}}(x) = \frac{(x-Q_{\mathrm{FP4}}(x))^2}{x^2}$.

2.2 Matrix-product error under RTN and SIC

Let $W \in \mathbb{R}^{m\times n}$ and $X \in \mathbb{R}^{n\times k}$, where the columns of X are calibration activations, and define $\Sigma_X = XX^\top/k$. We partition the inner dimension into $L = n/16$ groups: $W_\ell = W(:, 1+16(\ell-1):$

16ℓ) and $X_\ell = X(1{+}16(\ell{-}1) : 16\ell, :)$, with $A_\ell = \mathrm{diag}(\alpha_\ell^{(1)}, \ldots, \alpha_\ell^{(16)})$. Write $\Gamma_{W_\ell} = \left\{\gamma_{W_\ell}^{(i)}\right\}_{i=1}^{m}$ and $\Gamma_{X_\ell} = \left\{\gamma_{X_\ell}^{(j)}\right\}_{j=1}^{k}$ for the row and column scales. The matrices entering the FP4 quantizers are

$$W'_\ell = \mathrm{diag}(\Gamma_{W_\ell})^{-1} W_\ell A_\ell^{-1}, \qquad X'_\ell = A_\ell X_\ell \,\mathrm{diag}(\Gamma_{X_\ell})^{-1}. \tag{6}$$

For either deterministic or randomized quantization, let the dequantized matrices $\hat{W}_\ell$ and $\hat{X}_\ell$ approximate $W_\ell A_\ell^{-1}$ and $A_\ell X_\ell$, respectively. Denote their errors by e_{W_ℓ} and e_{X_ℓ} and concatenate them into e_W and e_X. The matrix-product error is

$$e = \hat{W}\hat{X} - WX = e_W AX + WA^{-1}e_X + e_W e_X \approx e_W AX + WA^{-1}e_X. \tag{7}$$

Here we neglect the second-order term $e_W e_X$.

To construct the optimization loss, we now analyze the randomized quantizer $\tilde{Q}_{\mathrm{FP4}}$ from (3); the final quantization procedure, however, uses the deterministic quantizer Q_{FP4}, as described in Section 3. We will optimize A, Γ_W, Γ_X in order to optimize the distortion of the randomized quantizer, then fix the obtained A, and afterwards refine Γ_W with respect to the deterministic quantizer. Since the randomized quantizer's relative MSE ϕ is minimized on $\mathcal{I}_{\mathrm{FP4}}$, the rough goal is to choose A, Γ_W, Γ_X so that as many entries as possible of the matrices in (6) have magnitudes in this interval.

In accordance with Lemma 1, we model the randomized FP4 errors as

$$\tilde{Q}_{\mathrm{FP4}}(y) = y + \sqrt{y^2\phi(y)}Z, \tag{8}$$

where Z has zero mean and unit variance, with independent noise across entries and blocks. These are modeling assumptions, rather than exact unbiasedness and independence properties of the quantization errors. The complete derivations are given in Appendix I.

Round to nearest (RTN) quantization: To analyze RTN, we apply the randomized quantizer to the scaled signal without feedback, and multiply the output by the inverse scaling. Specifically, given A, Γ_W under RTN quantization the entries of the weight matrix W are quantized to

$$\hat{W}_\ell^{\mathrm{FP4}} = \tilde{Q}_{\mathrm{FP4}}\left(\mathrm{diag}(\Gamma_{W_\ell})^{-1} W_\ell A_\ell^{-1}\right), \;\text{ so that }\; \hat{W}_\ell = \mathrm{diag}(\Gamma_{W_\ell})\tilde{Q}_{\mathrm{FP4}}\left(\mathrm{diag}(\Gamma_{W_\ell})^{-1} W_\ell A_\ell^{-1}\right). \tag{9}$$

With the block scales $\left\{\gamma_{W_\ell}^{(i)}\right\}_{i=1}^{m}$ and smoothing coefficients $\left\{\alpha_\ell^{(r)}\right\}_{r=1}^{16}$ defined above, we obtain

$$e_{W_\ell}(i,r) = \hat{W}_\ell(i,r) - \frac{W_\ell(i,r)}{\alpha_\ell^{(r)}} = \sqrt{\left(\frac{W_\ell(i,r)}{\alpha_\ell^{(r)}}\right)^2 \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right)} Z_\ell(i,r), \tag{10}$$

where $\{Z_\ell(i,r)\}$ are iid zero-mean, unit-variance random variables.

Let

$$L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) = \mathbb{E}\|e_{W_\ell} A_\ell X_\ell\|_F^2, \tag{11}$$

where $X_\ell, W_\ell, \Gamma_{W_\ell}, A_\ell$ are assumed fixed and the expectation is only with respect to the randomization of $\tilde{Q}_{\mathrm{FP4}}(\cdot)$, that is, with respect to the iid random variables $\{Z_\ell(i,r)\}$. It immediately follows that

$$\begin{aligned} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) &= \sum_{j=1}^{k}\sum_{i=1}^{m}\sum_{r=1}^{16} W_\ell^2(i,r) X_\ell^2(r,j) \cdot \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right) \\ &= k\sum_{r=1}^{16}\sigma_{X_\ell}^2(r)\sum_{i=1}^{m} W_\ell^2(i,r)\cdot\phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right), \end{aligned} \tag{12}$$

where $\sigma_{X_\ell}^2(r) = \frac{1}{k}\sum_{j=1}^{k} X_\ell^2(r,j)$ (we also have $\sigma_{X_\ell}^2(r) = \Sigma_X(r+16(\ell-1), r+16(\ell-1))$).

We assume that the random variables $\{Z_\ell(\cdot,\cdot)\}$ corresponding to different blocks are also independent. This implies

$$L_W^{\mathrm{RTN}} = \mathbb{E}\|e_W AX\|_F^2 = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell). \tag{13}$$

If activations are also quantized, applying RTN with the randomized FP4 quantizer gives

$$\hat{X}_\ell^{\mathrm{FP4}} = \tilde{Q}_{\mathrm{FP4}}\left(A_\ell X_\ell \,\mathrm{diag}(\Gamma_{X_\ell})^{-1}\right), \quad \text{so that} \quad \hat{X}_\ell = \tilde{Q}_{\mathrm{FP4}}\left(A_\ell X_\ell \,\mathrm{diag}(\Gamma_{X_\ell})^{-1}\right)\mathrm{diag}(\Gamma_{X_\ell}), \tag{14}$$

and

$$e_{X_\ell}(r,j) = \hat{X}_\ell(r,j) - \alpha_\ell^{(r)} X_\ell(r,j) = \sqrt{\left(\alpha_\ell^{(r)} X_\ell(r,j)\right)^2 \phi\left(\frac{\alpha_\ell^{(r)}}{\gamma_{X_\ell}^{(j)}} X_\ell(r,j)\right)}\,\tilde{Z}_\ell(r,j), \tag{15}$$

where $\left\{\gamma_{X_\ell}^{(j)}\right\}_{j=1}^{k}$ denote the activation block scales, and $\{\tilde{Z}_\ell(r,j)\}$ are iid zero-mean unit-variance random variables statistically independent of everything. Similar to the derivation above, we obtain

$$L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell) = \mathbb{E}\|W_\ell A_\ell^{-1} e_{X_\ell}\|_F^2 = m \sum_{r=1}^{16} \sigma_{W_\ell}^2(r) \sum_{j=1}^{k} X_\ell^2(r,j) \cdot \phi\left(\frac{\alpha_\ell^{(r)}}{\gamma_{X_\ell}^{(j)}} X_\ell(r,j)\right), \tag{16}$$

where $\sigma_{W_\ell}^2(r) = \frac{1}{m}\sum_{i=1}^{m} W_\ell^2(i,r)$. We also have

$$L_X^{\mathrm{RTN}} = \mathbb{E}\|W A^{-1} e_X\|_F^2 = \sum_{\ell=1}^{L} L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell). \tag{17}$$

Successive interference cancellation (SIC): Note that

$$\|e_W A X\|_F^2 = \mathrm{tr}\left((e_W A X)(e_W A X)^\top\right) = k\,\mathrm{tr}\left(e_W A \Sigma_X A^\top e_W^\top\right). \tag{18}$$

Let $U \in \mathbb{R}^{n\times n}$ be an upper triangular matrix such that $\Sigma_X = U^\top U$, and denote

$$Y = UA^\top (WA^{-1})^\top = UW^\top \in \mathbb{R}^{n\times m}. \tag{19}$$

We have

$$\|e_W A X\|_F^2 = k\|UA^\top e_W^\top\|_F^2 = k\|Y - UA\hat{W}^\top\|_F^2. \tag{20}$$

Even when A and the E4M3 scaling matrices $\{\Gamma_{W_\ell}\}_{\ell=1}^{L}$ are fixed, the problem of choosing $\hat{W}^{\mathrm{FP4}} \in \mathcal{C}_{\mathrm{FP4}}^{m\times n}$ such as to minimize (20) is computationally difficult due to the coupling the matrix UA introduces between different columns of $\hat{W}$ (equivalently, rows of $\hat{W}^\top$). As common in the literature, instead of attempting an optimal solution, we resort to the SIC/GPTQ approach. Taking into account the structure of the NVFP4 constellation, this boils down to quantizing the weight columns successively, starting from the last, and feeding the error back to the remaining columns, as in Algorithm 2 in Appendix I.

The analysis of $L_W^{\mathrm{SIC}} = k\mathbb{E}\|Y - UA\hat{W}^\top\|_F^2$ under the SIC algorithm is a bit complicated, even under the randomized FP4 quantization error model (8). The reason is that Z entries from time t affect $Y^{(t')}$ for $t' < t$ due to the feedback nature of the quantization process. We therefore make a high-resolution quantization assumption and assume the energy of the (filtered) quantization noise is negligible next to the signal (entries of Y) energy. Under this assumption we have that $Y^{(t)}(t,:) \approx U(t,t)\cdot(W(:,t))^\top$. Under this additional approximation,

$$L_{W_\ell}^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) = k \sum_{r=1}^{16} U_\ell^2(r) \sum_{i=1}^{m} W_\ell^2(i,r)\phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right), \tag{21}$$

where $U_\ell(r) = U(16(\ell-1)+r, 16(\ell-1)+r)$. The only difference from $L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell)$ is replacing $\sigma_{X_\ell}^2(r)$ by $U_\ell^2(r)$. The latter is the variance of the optimal prediction error of coordinate $t = 16(\ell-1)+r$ from coordinates $t+1:n$ of a Gaussian vector with covariance Σ_X, and hence $U_\ell^2(r) \le \sigma_{X_\ell}^2(r)$. This distinction allows us to optimize A taking into account that SIC will be used for the actual quantization.

Total quadratic Distortion: Using (12) (16) and (21) we obtain the following loss functions for randomized NVFP4 quantization, as a function of A and the E4M3 scales, under the various quantization setups:

$$L^{W-\mathrm{RTN}} = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) \tag{22}$$

$$L^{W-\mathrm{SIC}} = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) \tag{23}$$

$$L^{W-\mathrm{RTN},X-\mathrm{RTN}} = \sum_{\ell=1}^{L} \left[L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) + L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell)\right] \tag{24}$$

$$L^{W-\mathrm{SIC},X-\mathrm{RTN}} = \sum_{\ell=1}^{L} \left[L_{W_\ell}^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) + L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell)\right]. \tag{25}$$

Note that in all expressions the dependency on A and the E4M3 scales *completely decouples to L different functions, that depend on different parameters*. Therefore, the optimization on $\{A_\ell, \Gamma_{W_\ell}, \Gamma_{X_\ell}\}$ decouples to L independent optimization problems.

3 The OSFP4 Quantization Algorithm

We are now ready to describe how the Optimized Smoothing and Scaling for NVFP4 (OSFP4) quantization algorithm works. This algorithm has four different configurations: $W-\mathrm{RTN}, W-\mathrm{SIC}, W-\mathrm{RTN}/X-\mathrm{RTN}$ or $W-\mathrm{SIC}/X-\mathrm{RTN}$, corresponding to either W4A16 or W4A4, and to whether W is quantized using RTN or using SIC.

The parameters for each linear projection WX are optimized independently. We describe the procedure for a single projection operation. Our quantization pipeline is as follows.

Calibration set: For all 4 configurations we first collect k_1 activation vectors $x_1, \ldots, x_{k_1}$ and set $\Sigma_X = \frac{1}{k_1}\sum_{j=1}^{k_1} x_j x_j^\top$. Since computing Σ_X does not require many resources even when k_1 is large, we typically use large k_1, as is standard in calibration based quantization. The terms $L^{W-\mathrm{RTN}}, L^{W-\mathrm{SIC}}$ depend on the calibration set only through Σ_X, and for weight-only quantization we therefore keep only Σ_X and set $k = k_1$.

In the setups involving activations quantization, the term $L^{X-\mathrm{RTN}}$ also participates in the loss function, and the complexity of optimizing this term grows with k. Even though the optimization is performed offline, we will limit k to modest values, such that the quantization procedure will not require too many resources. We therefore take $k < k_1$ and select a random subset $\mathcal{J} = \{j_1, \ldots, j_k\}$ of k calibration vectors and set $X = [x_{j_1}|\cdots|x_{j_k}] \in \mathbb{R}^{n\times k}$.

Optimization of A (and initial guess for Γ_W): Depending on the configuration $W-\mathrm{RTN}, W-\mathrm{SIC}, W-\mathrm{RTN}/X-\mathrm{RTN}$ or $W-\mathrm{SIC}/X-\mathrm{RTN}$ we construct the relevant loss function from (22)-(25) (note that the first two depend only on Σ_X, whereas the last two also depend on X). Then, for every $\ell = 1, \ldots, L$ we perform joint optimization on A_ℓ, Γ_{W_ℓ} and also Γ_{X_ℓ} whenever relevant. As described above, all loss functions (22)-(25) decouple to L independent terms, so optimizing for each ℓ separately is optimal. The optimization is iterative and is initialized with $A = I_n$ and the Γ_W, Γ_X corresponding to absmax-to-6 scaling. The precise details of the optimization iterations are given in Appendix B. To illustrate the benefit of this optimization over the standard initialization values, Figure 3 in Appendix B shows how much $L^{W-\mathrm{SIC},X-\mathrm{RTN}}$ improved from its initial value along the different layers of Llama-3.1-8B-Instruct.

The value of A found by optimization is the final value used for quantization. The value found for Γ_W will be used to dictate the search space for the next step. Even in the setups $W-\mathrm{RTN}/X-\mathrm{RTN}$ or $W-\mathrm{SIC}/X-\mathrm{RTN}$ The value of Γ_X is irrelevant for the actual quantization of the activations, and will be determined in real-time based on the values of the actual activations computed in inference.

Optimization of Γ_W: The value of Γ_W found in the previous step was optimized assuming the randomized quantizer $\tilde{Q}_{\mathrm{FP4}}$ is used, whose output is generally not in $\mathcal{C}_{\mathrm{FP4}}$. Since in practice we are

restricted to output $\hat{W}^{\mathrm{FP4}} \in \mathcal{C}_{\mathrm{FP4}}^{m\times n}$, the randomized quantizer is not applicable, and we should use $Q_{\mathrm{FP4}}(\cdot)$ instead. Nevertheless, using part 1 of Lemma 1, for any fixed $y \in \mathbb{R}^{16}$ and $\gamma_{W_\ell}^{(i)}$ there must be some $\beta_{W_\ell}^{(i)} \in [1,2)$ such that the distortion for the deterministic $\gamma_{W_\ell}^{(i)}\beta_{W_\ell}^{(i)} Q_{\mathrm{FP4}}\left(\frac{y}{\gamma_{W_\ell}^{(i)}\beta_{W_\ell}^{(i)}}\right)$ is no larger than the expected distortion of the randomized quantizer $\gamma_{W_\ell}^{(i)}\tilde{Q}_{\mathrm{FP4}}\left(\frac{y}{\gamma_{W_\ell}^{(i)}}\right)$. In other words, the effective deterministic scale $\gamma' = \gamma_{W_\ell}^{(i)}\beta_{W_\ell}^{(i)}$ lies in $[\gamma_{W_\ell}^{(i)}, 2\gamma_{W_\ell}^{(i)}]$, and the deterministic quantizer $\gamma' Q_{\mathrm{FP4}}\left(\frac{y}{\gamma'}\right)$ attains no larger distortion than the expected distortion of the randomized quantizer $\gamma_{W_\ell}^{(i)}\tilde{Q}_{\mathrm{FP4}}\left(\frac{y}{\gamma_{W_\ell}^{(i)}}\right)$. In fact, the distortion for the optimal γ' may be substantially smaller than that of the randomized quantizer, as it can always place at least one of the entries of y on a point in $\mathcal{C}_{\mathrm{FP4}}$, cancelling its quantization error completely.

Consequently, for each $\ell = 1,\ldots,L$ and $i = 1,\ldots,m$ we determine the final scale by trying all valid scale values in an interval containing $[\gamma_{W_\ell}^{(i)}, 2\gamma_{W_\ell}^{(i)}]$. We can only use scales in $\mathcal{C}_{\mathrm{scale}}$, the set of positive E4M3 numbers. Consequently, not all values in $[\gamma_{W_\ell}^{(i)}, 2\gamma_{W_\ell}^{(i)}]$ can be used, and we therefore choose the best scale in $\mathcal{C}_{\mathrm{scale}}^{+}(\gamma_{W_\ell}^{(i)}) = \mathcal{C}_{\mathrm{scale}} \cap \left[\frac{\gamma_{W_\ell}^{(i)}}{1.2}, \frac{\gamma_{W_\ell}^{(i)}}{0.3}\right]$. The fine-grained scale-selection procedure is described in Appendix C. Projection-level comparisons of the predicted loss, randomized quantization, and final deterministic scale selection for all four configurations are given in Appendix E.

Optimization of Γ_X: We first need to multiply the activation X by the diagonal matrix A. This is handled in the LayerNorm whenever it exists, or via an additional diagonal scaling layer when there is no LayerNorm. Under $W-\mathrm{RTN}$ and $W-\mathrm{SIC}$ the activation X is not quantized, whereas under $W-\mathrm{RTN}/X-\mathrm{RTN}$ and $W-\mathrm{SIC}/X-\mathrm{RTN}$ the activations also need to be quantized. Unlike the quantization of W which is done offline, the quantization of X is done in inference time and should therefore be sufficiently fast. Our default is to compute a global scale for X based on the calibration set, and then quantize each 16 consecutive entries using absmax-to-6. The fixed global scale allows to perform this quantization online, even before the entire tensor of activations was computed. This is the default choice in our experimental results. We nevertheless mention that recent work Gupta et al. (2026); Lin & Wan (2026) advocates searching a small bank of E4M3 values around the absmax-to-6 value even for the activations, and shows that the overhead this search incurs on the quantization time is quite reasonable (Gupta et al., 2026, Table 6). This optimization is referred to as ScaleSearch/ScaleSweep.

4 Experimental Results

We evaluate whether OSFP4 improves NVFP4 accuracy, how its components contribute, and whether these gains retain the efficiency of native FP4 inference. We focus here on Llama-3.1-8B-Instruct; Appendix F contains results for additional models. Locally prepared models share the same source checkpoint and calibration set; all models are evaluated through vLLM on an RTX 5090. We report W4A4 and W4A16 separately and distinguish absmax activation scaling from methods that search activation scales during inference, which typically incur larger overhead. Full task results, calibration settings, and implementation details are given in Appendix D.

Accuracy under a common evaluation protocol. Table 1 summarizes the mean accuracy on MMLU-CoT, GSM8K, HellaSwag, and WinoGrande. With absmax activation scaling, OSFP4 W-SIC/X-RTN achieves 77.03%, compared with 76.34% for FP-Quant GPTQ and 76.18% for MR-GPTQ, and has the highest average among the evaluated methods in this group. The weight-only configuration reaches 78.42%, versus 78.00% for H-Scale and 79.22% for BF16. Thus, the accuracy improvement is present both with and without activation quantization.

Online activation-scale search is complementary to OSFP4, and was evaluated mainly for enabling a fair comparison with ScaleSweep Lin & Wan (2026). Note that even with absmax activation scaling

Table 1: Mean accuracy (%) on Llama-3.1-8B-Instruct. Full scores: Table 3.

Method	Avg.
BF16 baseline	
BF16	**79.22**
W4A4: absmax activation scaling	
RTN (FP-Quant)	75.65
GPTQ (FP-Quant)	76.34
MR-GPTQ (FP-Quant)	76.18
SOAR	76.60
H-Scale (W4A4 extension)	76.67
NVIDIA released checkpoint†	76.11
OSFP4 W-RTN / X-RTN	76.36
OSFP4 W-SIC / X-RTN	**77.03**
W4A4: online activation-scale search	
ScaleSweepMSE	76.62
ScaleSweepMSE + GPTQ	76.69
4over6	76.34
OSFP4 W-RTN / X-ScaleSweepMSE	76.96
OSFP4 W-SIC / X-ScaleSweepMSE	**77.09**
W4A16: weights only	
RTN	77.67
H-Scale	78.00
OSFP4 W-RTN	78.15
OSFP4 W-SIC	**78.42**

†Separately released checkpoint; preparation is uncontrolled. Bold: best quantized average in each group.

Table 2: WikiText-2 perplexity (↓). W4A4 uses absmax activation scales.

Configuration	W4A16	W4A4
BF16 reference	7.22	7.22
Absmax RTN ($A = I$)	7.55	7.88
MSE scales ($A = I$)	7.52	7.85
OSFP4 W-RTN	7.42	7.77
SIC ($A = I$, absmax)	7.45	7.77
OSFP4 W-SIC	**7.39**	**7.67**

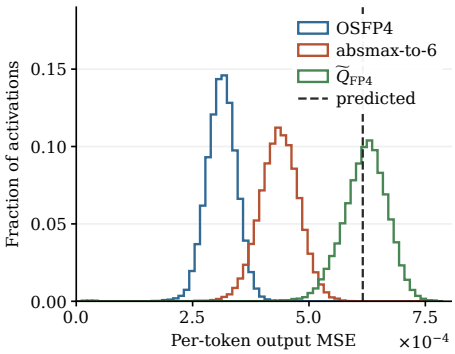


Figure 2: Layer 12:value, W-RTN/W4A16. Full three-projection comparison: Figure 4.

OSFP4 W-SIC/X-RTN scores better than ScaleSweepMSE+GPTQ, whereas the latter does optimize the activation scale online.

Contribution of smoothing, scale selection, and SIC. Table 2 compares the full procedure with simpler configurations. Optimizing only the weight scales by unweighted MSE provides a modest perplexity improvement over absmax. OSFP4 W-RTN optimizes A and selects the weight scales using weighted MSE yielding a more significant improvement. Combining the optimized transformation and scale selection with SIC gives the lowest perplexity among the NVFP4 configurations in both precision settings. In W4A4, perplexity decreases from 7.88 for absmax RTN to 7.67 for OSFP4 W-SIC, compared with 7.77 for SIC using $A = I$ and absmax scales. Appendix H further details an ablation study of the MSE for projection-output error across all seven projections, for all variants. The full method (optimized A, scales and SIC) attains the smallest MSE across all layers and compared schemes. These configurations assess the combined procedure rather than isolating the contribution of A alone. Figure 3 in Appendix B shows the reduction in the joint optimization objective across projections.

From predicted loss to deterministic quantization. Figure 2 examines the two-stage procedure on the value projection in layer 12. For W-RTN/W4A16, the mean error of the randomized quantizer agrees with the predicted loss $L^{W-\mathrm{RTN}}(A^*, \Gamma_W^*)$ to within 1%. The final deterministic E4M3 scale search reduces mean error to approximately half the prediction and improves over absmax weight scaling at the same optimized A. This supports using the smoothed loss to guide optimization, followed by fine-grained deterministic scale selection. Appendix E provides details and more extensive results in this line.

Inference efficiency. On an NVIDIA B200, OSFP4 retains approximately 94–97% of the vendor NVFP4 prefill throughput across the measured workloads. This modest gap is primarily attributable to the explicit diagonal smoothing required before the attention output and MLP down projections, where the transformation cannot be absorbed into a preceding LayerNorm. Consistent with this explanation, a control that sets $A = I$ while retaining the same runtime path achieves essentially the same throughput as OSFP4. Appendix G reports the full workload sweep and timing protocol.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with writing and exposition, for writing code and running experiments, and for assistance in implementation of the scale optimization algorithm, as described in Appendix B. We have not used generative AI tools for any original idea. The conceptualization of this project, the idea to use the dithered quantizer for developing a convenient loss function, and the mathematical development of the loss function are solely due to the human authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide a complete reproduction package for the OSFP4 experiments reported in this paper. It contains the experiment scripts, configurations, software dependency versions, GPU specifications, saved measurements, and commands for regenerating the reported tables and figures. The package is available in an OSFP4 reproduction repository.

REFERENCES

Anonymous. SmoothGPTQ: Optimizing diagonal scaling for GPTQ weight-activation quantization. Submitted to ICLR 2027; under review, 2027.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.00456.

Chengzhu Bao, Xianglong Yan, Zhiteng Li, Guangshuo Qin, Guanghua Yu, and Yulun Zhang. SOAR: Scale optimization for accurate reconstruction in NVFP4 quantization. *arXiv preprint arXiv:2605.12245*, 2026. URL https://arxiv.org/abs/2605.12245.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2307.13304.

Jiale Chen, Vage Egiazarian, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. Wush: Near-optimal adaptive transforms for llm quantization. *arXiv preprint arXiv:2512.00956*, 2025.

Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, Keith Wyss, Mahdi Nazemi, Asit Mishra, Carlo del Mundo, Tijmen Blankevoort, and Song Han. Four over six: More accurate NVFP4 quantization with adaptive block scaling. *arXiv preprint arXiv:2512.02010*, 2026. URL https://arxiv.org/abs/2512.02010.

Vage Egiazarian, Roberto Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling fp4 quantization. In *International Conference on Learning Representations*, volume 2026, pp. 113529–113563, 2026.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

Tanmaey Gupta, Hayden Prairie, Xiaoxia Wu, Reyna Abhyankar, Qingyang Wu, Austin Silveria, Pragaash Ponnusamy, Jue Wang, Ben Athiwaratkun, Leon Song, Tri Dao, Daniel Y. Fu, and Chris De Sa. Search your block floating point scales! *Proceedings of Machine Learning and Systems*, 8:1383–1398, 2026. URL https://arxiv.org/abs/2605.12464.

Egor Lifar, Semyon Savkin, Or Ordentlich, and Yury Polyanskiy. WaterSIC: Information-theoretically (near) optimal linear layer quantization. *arXiv preprint arXiv:2603.04956*, 2026. URL https://arxiv.org/abs/2603.04956.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024. URL https://arxiv.org/abs/2306.00978.

Li Lin and Xiaojun Wan. ScaleSweep: Accurate NVFP4 post-training quantization of LLMs via block scale initialization. *arXiv preprint arXiv:2606.07618*, 2026. URL https://arxiv.org/abs/2606.07618.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2405.16406.

Or Ordentlich and Yury Polyanskiy. High-rate quantized matrix multiplication i. *IEEE BITS the Information Theory Magazine*, 2026a.

Or Ordentlich and Yury Polyanskiy. High-rate quantized matrix multiplication II. *arXiv preprint arXiv:2605.13768*, 2026b. URL https://arxiv.org/abs/2605.13768.

Semyon Savkin, Eitan Porat, Or Ordentlich, and Yury Polyanskiy. Nestquant: Nested lattice quantization for matrix products and llms. *arXiv preprint arXiv:2502.09720*, 2025.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.13137.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2410.09426.

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

Hao Yu, Zheng Li, Dayiheng Liu, and Jianwei Zhang. H-Scale: Hessian-guided scale refinement for NVFP4 sub-byte LLM inference. *arXiv preprint arXiv:2608.28113*, 2026. URL https://arxiv.org/abs/2608.28113.

A Proof of Lemma 1

We first express the smoothed relative error as an integral over a unit interval in logarithmic coordinates.

Proposition 1 (Integral representation of the smoothed relative error) *For every $t \in \mathbb{R}$,*

$$\phi(2^t) = \int_{t-1}^{t} g(v)\, dv, \qquad g(v) = \left(1 - 2^{-v} Q_{\mathrm{FP4}}(2^v)\right)^2. \tag{26}$$

Proof. Substituting (3) into the definition of ϕ and using $U \sim \mathrm{Uniform}([0,1))$ gives

$$\begin{aligned} \phi(2^t) &= \mathbb{E}_U\left[\left(1 - 2^{U-t} Q_{\mathrm{FP4}}(2^{t-U})\right)^2\right] \\ &= \int_0^1 \left(1 - 2^{u-t} Q_{\mathrm{FP4}}(2^{t-u})\right)^2 du. \end{aligned} \tag{27}$$

The substitution $v = t - u$, with $du = -dv$, yields (26). ■

Proof of Lemma 1. The first part follows since the minimum is upper bounded by the expectation, taking $\beta = 2^U$. The fact that $\phi(-x) = \phi(x)$ follows immediately from $Q_{\mathrm{FP4}}(-x) = -Q_{\mathrm{FP4}}(x)$. It

remains to show that $\phi(x) \geq \phi_{\min}$ and that this minimum is attained for all $x \in \mathcal{I}_{\mathrm{FP4}}$. To that end, we first resort to the infinite exponent model of Ordentlich & Polyanskiy (2026a, Section III.B). In particular, define the "$E\infty M1$" constellation

$$C_{E\infty M1} = \{0\} \cup \left\{\pm 2^k,\ \pm\tfrac{3}{2}\,2^k : k \in \mathbb{Z}\right\}, \tag{28}$$

which is clearly a superset of $\mathcal{C}_{\mathrm{FP4}}$. Defining

$$Q_\infty(x) = \operatorname*{argmin}_{\hat{x} \in \mathcal{C}_{E\infty M1}} |x - \hat{x}|, \tag{29}$$

it therefore follows that

$$\begin{aligned} g_\infty(v) &\triangleq \left(1 - 2^{-v} Q_\infty(2^v)\right)^2 = 2^{-2v}\left(2^v - Q_\infty(2^v)\right)^2 \\ &\leq 2^{-2v}\left(2^v - Q_{\mathrm{FP4}}(2^v)\right)^2 = g(v), \quad \forall v \in \mathbb{R}. \end{aligned} \tag{30}$$

Consequently

$$\phi(2^t) = \int_{t-1}^{t} g(v)\,dv \geq \int_{t-1}^{t} g_\infty(v)\,dv \overset{(a)}{=} \int_1^2 g_\infty(v)\,dv \overset{(b)}{=} \phi_{\min}, \tag{31}$$

where in (a) we have used the fact that $2^{-(v+z)}Q_\infty(2^{(v+z)}) = 2^{-v}Q_\infty(2^v)$ for all $v \in \mathbb{R}$ and $z \in \mathbb{Z}$ (since $2^z C_{E\infty M1} = C_{E\infty M1}$). We now establish equality (b) above . This follows since for any $x \in [2, 4)$ we have that $Q_\infty(x) = \mathrm{round}(x)$, which implies also that for any $1 \leq v < 2$ it holds that $g_\infty(v) = (1 - 2^{-v}\mathrm{round}(2^v))^2$. Consequently

$$\int_1^2 g_\infty(v)\,dv = \int_1^2 (1 - 2^{-v}\mathrm{round}(2^v))^2\,dv = \int_0^1 (1 - 2^{-(u+1)}\mathrm{round}(2^{(u+1)}))^2\,du = \phi_{\min} \tag{32}$$

as required.

The bound (31) holds with equality whenever (30) holds with equality. Equality in (30) in turn holds for all v for which $Q_\infty(2^v) = Q_{\mathrm{FP4}}(2^v)$. Noting that

$$C_{E\infty M1} = \{\ldots, \tfrac{3}{4},\ 1,\ \tfrac{3}{2},\ 2,\ 3,\ 4,\ 6,\ 8, \ldots\} \tag{33}$$

we see that

$$Q_\infty(y) = Q_{\mathrm{FP4}}(y) \quad \forall y \in [7/8, 7]. \tag{34}$$

In particular, for any $x \in \mathcal{I}_{\mathrm{FP}} = [7/4, 7]$ we have that $Q_\infty(x) = Q_{\mathrm{FP4}}(x)$ on the entire interval $[x/2, x]$. Consequently, setting $t_x = \log_2(x)$ we see that for any $x \in \mathcal{I}_{\mathrm{FP}} = [7/4, 7]$ it holds that $g(v) = g_\infty(v)$ for all $v \in [t_x - 1, t_x]$ and therefore (30) holds with equality for all $x \in \mathcal{I}_{\mathrm{FP}} = [7/4, 7]$, as claimed. ■

B Implementation of the Scale Optimization

This appendix specifies the continuous optimization of A, Γ_W, and, when activations are quantized, Γ_X. It also describes how the resulting parameters determine the FP32 tensor scales used for deployment. The continuous optimization and the subsequent selection of E4M3 weight scales are separate steps: the former uses the smoothed loss ϕ, whereas the latter evaluates deterministic FP4 quantization error.

Calibration statistics and parameterization. For a weight matrix $W \in \mathbb{R}^{m\times n}$, let k_1 be the number of calibration activation vectors used to estimate Σ_X. For W4A4, the activation term uses a fixed subset of $k \leq k_1$ complete activation vectors, sampled uniformly without replacement. This subset is selected once for the mapping and reused in every optimization iteration. Weight-only optimization requires Σ_X but does not retain an activation matrix in its objective. When several projections share one smoothing transformation, their weight matrices are stacked along the output dimension and optimized as one mapping; m then denotes the combined number of output rows.

For each group ℓ, the trainable parameters are the natural logarithms

$$\begin{aligned} a_{\ell r} &= \log \alpha_\ell^{(r)}, & r &= 1, \dots, 16, \\ b_{\ell i} &= \log \gamma_{W_\ell}^{(i)}, & i &= 1, \dots, m, \\ c_{\ell j} &= \log \gamma_{X_\ell}^{(j)}, & j &= 1, \dots, k. \end{aligned} \tag{35}$$

The $c_{\ell j}$ parameters are omitted for W4A16. Scales are reconstructed by exponentiation, ensuring positivity. We initialize $\alpha_\ell^{(r)} = 1$, $\gamma_{W_\ell}^{(i)} = \max_r |W_\ell(i,r)|/6$, and $\gamma_{X_\ell}^{(j)} = \max_r |X_\ell(r,j)|/6$, with positive numerical floors for zero-valued blocks. The weights, calibration activations, and second-moment statistics are held fixed throughout this optimization. The loss and trainable log parameters use FP32 arithmetic.

Objective and normalization. Define the fixed weight metric

$$d_{\ell r} = \begin{cases} \Sigma_X(t,t), & \text{for weight RTN,} \\ U(t,t)^2, & \text{for weight SIC,} \end{cases} \qquad t = 16(\ell - 1) + r. \tag{36}$$

For robustness, the implementation computes the SIC factor from the damped second moment,

$$U^\top U = \Sigma_X + \lambda I_n, \qquad \lambda = 0.01 \, \frac{\operatorname{tr}(\Sigma_X)}{n}, \tag{37}$$

as is common in GPTQ. The per-group losses evaluated in the implementation are

$$\mathcal{J}_{W_\ell} = \frac{1}{m} \sum_{r=1}^{16} \sum_{i=1}^{m} d_{\ell r} W_\ell(i,r)^2 \phi\left(\frac{W_\ell(i,r)}{\alpha_\ell^{(r)} \gamma_{W_\ell}^{(i)}} \right), \tag{38}$$

$$\mathcal{J}_{X_\ell} = \frac{1}{mk} \sum_{r=1}^{16} \sum_{j=1}^{k} \left(\sum_{i=1}^{m} W_\ell(i,r)^2 \right) X_\ell(r,j)^2 \phi\left(\frac{\alpha_\ell^{(r)} X_\ell(r,j)}{\gamma_{X_\ell}^{(j)}} \right). \tag{39}$$

Thus, the scalar objective is $\mathcal{J} = \sum_\ell \mathcal{J}_{W_\ell}$ for W4A16 and $\mathcal{J} = \sum_\ell (\mathcal{J}_{W_\ell} + \mathcal{J}_{X_\ell})$ for W4A4. These are the corresponding losses in Section 2, normalized by mk, with $\{d_{\ell r}\}$ in the weight metric estimated using all k_1 calibration vectors and the activation term averaged over the selected k vectors. The coefficients $d_{\ell r} W_\ell(i,r)^2$ and $X_\ell(r,j)^2 \sum_i W_\ell(i,r)^2$ are computed once. All groups are evaluated together in a batched computation; their parameters are disjoint, so summing their objectives introduces no coupling between groups.

Differentiable evaluation of ϕ. We evaluate the smoothed relative error through a lookup table indexed by $t = \log_2 |x|$. The table has spacing $h = 2^{-12}$ on $[-4, 8]$, with entries $\phi(2^{t_q})$ at $t_q = -4 + qh$. Each entry is computed using the identity proved in Proposition 1, by analytically integrating

$$\phi(2^t) = \int_{t-1}^{t} \left(1 - 2^{-v} Q_{\mathrm{FP4}}(2^v)\right)^2 \, dv, \tag{40}$$

splitting the integral at the FP4 decision boundaries. Table construction uses double precision and the cached table used by the optimizer is FP32. Values between adjacent entries are obtained by linear interpolation in t. Recall that the trainable variables are the log-scales a, b, c in (35). For nonzero inputs, t is affine in these variables, so their gradients follow directly from the interpolation slope with respect to t via the chain rule, with a factor of $1/\ln 2$ converting natural logarithms to base two. Automatic differentiation treats the integer cell index as fixed and differentiates through the interpolation fraction, the logarithm of the magnitude, and the exponential scale parameterization. Consequently, optimization uses the derivative of the interpolated surrogate, without sampling the randomized quantizer or differentiating through deterministic FP4 rounding. The implementation clamps the logarithmic argument to $[-4, 8 - h]$ before interpolation; outside this interval the lookup contribution has zero derivative with respect to its argument. At $x = 0$ the lookup is finite and its loss coefficient is zero.

Adam iterations. We jointly update all the log parameters using Adam for 80 iterations with a fixed learning rate $\eta = 0.12$, moment coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$, and $\epsilon = 10^{-8}$. Weight decay is zero and AMSGrad is disabled. Let θ_s collect the log parameters a, b, c after iteration s, with c omitted for weight-only quantization, and let $g_s = \nabla_\theta \mathcal{J}(\theta_{s-1})$. Adam maintains exponentially weighted moving averages p_s of the gradients and v_s of their elementwise squares. The updates are

$$\begin{aligned} p_s &= \beta_1 p_{s-1} + (1-\beta_1) g_s, & v_s &= \beta_2 v_{s-1} + (1-\beta_2) g_s^{\odot 2}, \\ \widehat{p}_s &= p_s/(1-\beta_1^s), & \widehat{v}_s &= v_s/(1-\beta_2^s), \\ \theta_{s,j} &= \theta_{s-1,j} - \eta \, \frac{\widehat{p}_{s,j}}{\sqrt{\widehat{v}_{s,j}} + \epsilon}, \qquad j = 1, \ldots, \dim(\theta), \end{aligned} \tag{41}$$

where j indexes the trainable log parameters, $p_0 = v_0 = 0$, and all operations are elementwise. The quantities $\widehat{p}_s$ and $\widehat{v}_s$ correct the moving averages for their zero initialization. Each iteration clears the previous gradients, evaluates the entire fixed objective, backpropagates, and updates the parameters simultaneously. There is no alternating minimization, learning-rate schedule, gradient clipping, or early stopping. The final iterate is retained, rather than the iterate with the smallest observed loss. The procedure seeks a low-loss solution of the nonconvex surrogate and does not impose monotonic descent.

Scale normalization and deployment precision. The objective is invariant under a common rescaling within each group. After the last Adam update, we exploit this freedom by setting

$$\begin{aligned} s_\ell &= \exp\left(\frac{1}{m} \sum_{i=1}^{m} \log \gamma_{W_\ell}^{(i)} \right), \\ \alpha_\ell^{(r)} &\leftarrow s_\ell \alpha_\ell^{(r)}, \qquad \gamma_{W_\ell}^{(i)} \leftarrow \gamma_{W_\ell}^{(i)} / s_\ell, \qquad \gamma_{X_\ell}^{(j)} \leftarrow s_\ell \gamma_{X_\ell}^{(j)}. \end{aligned} \tag{42}$$

This sets the geometric mean of the continuous weight scales to one while preserving both $W_\ell(i,r)/(\alpha_\ell^{(r)} \gamma_{W_\ell}^{(i)})$ and $\alpha_\ell^{(r)} X_\ell(r,j)/\gamma_{X_\ell}^{(j)}$. The normalization is applied once, after optimization. We then round the diagonal entries of A to the weight dtype when the transformation is folded into a preceding normalization layer, or to BF16 when explicit runtime smoothing is required. The subsequent deterministic E4M3 weight-scale selection uses this rounded A. Rounding and discrete scale selection are not part of the Adam iterations.

FP32 tensor scales. The global tensor scales are distinct from the block scales optimized above and are not additional Adam parameters. For weights, the saved FP32 global scale is explicitly set to one. After the normalization in (42), the weight-scale search selects E4M3 values directly, with the compensating scale already incorporated into A.

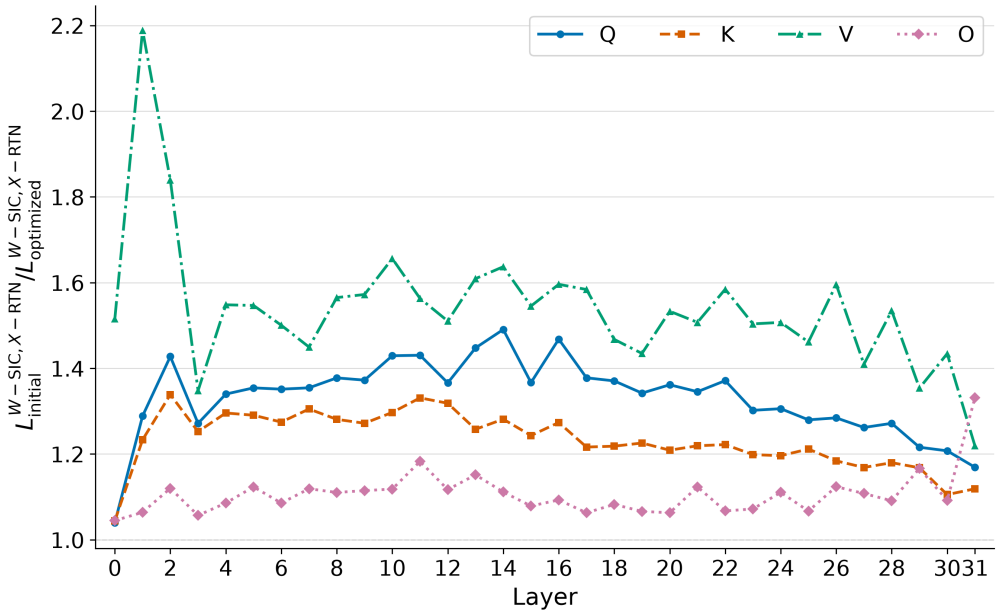
For W4A4 activations, the optimized Γ_X is used only to guide the choice of A and is discarded for deployment. After applying the final rounded diagonal transformation $A_\star$, we recompute the calibration maximum

$$M_X = \max_{1 \le j \le k_1} \|A_\star x_j\|_\infty. \tag{43}$$

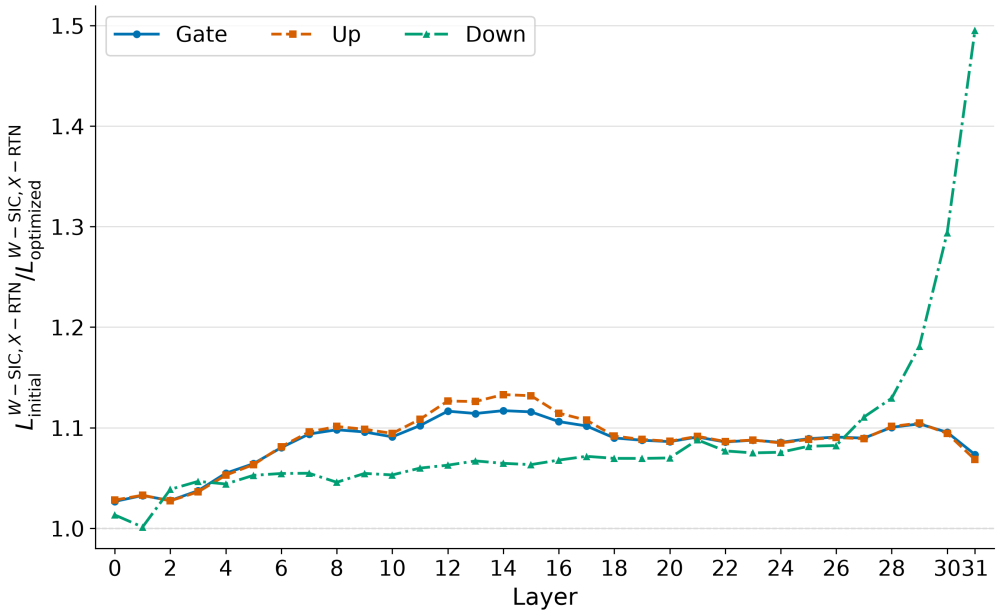
This uses the full calibration statistics, not only the subset used in the activation loss. The maximum can be recovered exactly from stored per-channel absolute maxima, since the diagonal scaling is fixed. The saved FP32 global *quantization multiplier* is

$$g_X = \frac{448 \cdot 6}{M_X}, \qquad s_X = g_X^{-1} = \frac{M_X}{448 \cdot 6}, \tag{44}$$

where 448 and 6 are the maximum positive E4M3 and E2M1 values, respectively, and s_X denotes the corresponding dequantization scale. Numerical safeguards replace nonfinite multipliers by one. This global scale remains fixed during inference. The E4M3 block scales are computed dynamically from each actual group of 16 smoothed activation entries using absmax-to-6 with this global multiplier. Weight-only configurations do not require activation quantization scales.



(a) Loss improvement for attention projections.



(b) Loss improvement for MLP projections

Figure 3: Ratio between $L^{W-\mathrm{SIC},X-\mathrm{RTN}}$ with default initialized values, and the optimized value, along the 32 layers of Llama-3.1-8B-instruct. The ratio is plotted for each one of the 7 projections.

C Detailed Fine-Grained E4M3 Scale-Selection Procedure

After the continuous optimization and scale normalization in Appendix B, we choose the final weight scales from the candidate sets $\mathcal{C}^{+}_{\mathrm{scale}}(\gamma^{(i)}_{W_\ell})$ defined in the main text. For each normalized continuous scale $\gamma = \gamma^{(i)}_{W_\ell}$, the implementation searches representable E4M3 candidate scales γ' in $[\gamma/1.2, \gamma/0.3]$. The rounded smoothing coefficients $\alpha^{(r)}_{\ell}$ are held fixed throughout the search.

Denote by $\Gamma_{W_\ell}^{\text{final}} = \text{diag}(\gamma_{W_\ell}^{\text{final},(1)}, ..., \gamma_{W_\ell}^{\text{final},(m)})$ the final E4M3 scales used for representing W_ℓ in NVFP4 format. For the $W-\text{RTN}$ and $W-\text{RTN}/X-\text{RTN}$ setups we set

$$\gamma_{W_\ell}^{\text{final},(i)} = \underset{\gamma \in \mathcal{C}^+_{\text{scale}}(\gamma_{W_\ell}^{(i)})}{\text{argmin}} \sum_{r=1}^{16} \sigma^2_{X_\ell}(r) \left(W_\ell(i,r) - \gamma \cdot \alpha_\ell^{(r)} \cdot Q_{\text{FP4}} \left(\frac{W_\ell(i,r)}{\gamma \cdot \alpha_\ell^{(r)}} \right) \right)^2, \quad \forall i = 1, \ldots, m, \tag{45}$$

and then set

$$\hat{W}_\ell^{\text{FP4}}(i,:) = Q_{\text{FP4}} \left(\frac{W_\ell(i,:) A_\ell^{-1}}{\gamma_{W_\ell}^{\text{final},(i)}} \right). \tag{46}$$

In the $W-\text{SIC}$ and $W-\text{SIC}/X-\text{RTN}$ setups the dependence of the MSE on $\Gamma_{W_\ell}^{\text{final}}$ is more involved because of the feedback nature of the SIC quantization algorithm. Under RTN, the choice of γ affects the quantization errors corresponding to 16 entries in the corresponding block in an independent manner, and is therefore optimized by (45). On the other hand, under SIC, errors in quantizing earlier coordinates propagate to the quantization of future coordinates by the SIC feedback loop. Consequently, optimizing $\left\{\Gamma_{W_\ell}^{\text{final}}\right\}_{\ell=1}^{L}$ is less straightforward. We use Algorithm 1 in order to choose $\left\{\Gamma_{W_\ell}^{\text{final}}\right\}_{\ell=1}^{L}$, and quantize W in the same run. The main challenge in choosing γ is that we do not know what will be the input to the quantizer before the feedback quantization procedure was executed. Algorithm 1 chooses the scale γ block by block, and relies on the decisions on $\hat{W}$ that were already committed in the previous blocks. When optimizing the scale for block ℓ, it further *estimates* the effect of intra-block feedback-filtered quantization by approximating $\hat{W}_\ell = W_\ell$ only for the current block.

Algorithm 1 NVFP4 SIC with Optimized E4M3

Inputs: $Y \in \mathbb{R}^{n \times m}$, upper triangular $U \in \mathbb{R}^{n \times n}$, diagonal matrices $\{A_\ell\}_{\ell=1}^{L=n/16}$ with $A_\ell = \text{diag}\left(\alpha_\ell^{(1)}, \ldots, \alpha_\ell^{(16)}\right)$, and initial diagonal scaling matrices $\{\Gamma_{W_\ell}\}_{\ell=1}^{L=n/16}$ with $\Gamma_{W_\ell} = \text{diag}\left(\gamma_{W_\ell}^{(1)}, \ldots, \gamma_{W_\ell}^{(m)}\right)$

Outputs: $W_{\text{SIC}} \in \mathcal{C}_{\text{FP4}}^{n \times m}$ and scale matrices $\{\Gamma_{W_\ell}^{\text{final}}\}_{\ell=1}^{L=n/16}$

$Y^{(n)} \leftarrow Y$
for $\ell = L : 1$ **do**
  $\mathcal{I}_\ell = \{1 + 16(\ell-1), \ldots, 16\ell\}$ ▷ The set of 16 indices participating in ℓth block
  $U_{\text{block}} = U(\mathcal{I}_\ell, \mathcal{I}_\ell)$ ▷ The ℓth 16×16 block in the diagonal of U
  $U_{\text{block}}^{\text{causal}} = U_{\text{block}} - \text{diag}(U_{\text{block}})$ ▷ Making U_{block} strictly upper triangular
  $Y_{\text{block}} \leftarrow Y^{(16\ell)}(\mathcal{I}_\ell, :)$
  $Y_{\text{block}}^{\text{approx}} \leftarrow Y_{\text{block}} - U_{\text{block}}^{\text{causal}} W_\ell^\top$ ▷ Approximating input to the quantizer before applying feedback iteration
  for $i = 1 : m$ **do**
    $\gamma_{W_\ell}^{\text{final},(i)} \leftarrow \text{argmin}_{\gamma \in \mathcal{C}^+_{\text{scale}}(\gamma_{W_\ell}^{(i)})}$
      $\sum_{a=1}^{16} \left(Y_{\text{block}}^{\text{approx}}(a,i) - \gamma \alpha_\ell^{(a)} U_{\text{block}}(a,a) \cdot Q_{\text{FP4}} \left(\frac{Y_{\text{block}}^{\text{approx}}(a,i)}{\gamma \alpha_\ell^{(a)} U_{\text{block}}(a,a)} \right) \right)^2$
  end for
  $\Gamma_{W_\ell}^{\text{final}} = \text{diag}(\gamma_{W_\ell}^{\text{final},(1)}, ..., \gamma_{W_\ell}^{\text{final},(m)})$
  for $r = 16 : 1$ **do**
    $t \leftarrow 16(\ell-1) + r$
    $W_{\text{SIC}}(t,:) \leftarrow Q_{\text{FP4}} \left(\frac{Y^{(t)}(t,:)\left(\Gamma_{W_\ell}^{\text{final}}\right)^{-1}}{\alpha_\ell^{(r)} U(t,t)} \right)$ ▷ For optimization use $\tilde{Q}_{\text{FP4}}$, for actual quantization use Q_{FP4}
    $Y^{(t-1)} \leftarrow Y^{(t)} - \alpha_\ell^{(r)} U(:,t) \cdot W_{\text{SIC}}(t,:) \Gamma_{W_\ell}^{\text{final}}$ ▷ $U(:,t)$ is the tth column of U
  end for
end for

D NATIVE NVFP4 COMPARISON: EXPERIMENTAL SETUP

Table 3 gives the full task results underlying Table 1. All 18 rows share one evaluation protocol. We distinguish W4A4 methods using absmax activation scaling, W4A4 methods searching activation scales online, and weight-only W4A16 methods. Bold entries indicate the best score within each group.

Table 3: Native NVFP4 accuracy on Llama-3.1-8B-Instruct under the common protocol in Appendix D. All entries are percentages; recovery is relative to the measured BF16 baseline. Bold indicates the best score within each group. FP-Quant rows use author-provided quantization with our native serving adapter; H-Scale W4A4 is our extension. †NVIDIA's separately released checkpoint has uncontrolled preparation. The online-search group includes 4over6, which selects activation scales by reconstruction error during inference. OSFP4 activation-ScaleSweep rows retain their corresponding X-RTN weights, A, and tensor scales.

Precision	Method	MMLU-CoT	GSM8K	HellaSwag	WinoGrande	Avg.	Recovery %
Baseline	BF16	72.77	85.97	80.41	77.74	**79.22**	**100.00**
Weights and activations: absmax activation scaling							
W4A4	RTN (FP-Quant)	68.61	80.29	78.87	74.82	75.65	95.48
W4A4	GPTQ (FP-Quant)	68.86	80.67	78.78	**77.03**	76.34	96.35
W4A4	MR-GPTQ (FP-Quant)	**69.84**	81.20	78.16	75.53	76.18	96.16
W4A4	SOAR	69.14	82.11	78.85	76.32	76.60	96.69
W4A4	H-Scale (W4A4 extension)	69.74	81.35	79.03	76.56	76.67	96.78
W4A4	NVIDIA released checkpoint†	68.49	80.89	78.75	76.32	76.11	96.07
W4A4	OSFP4 W-RTN / X-RTN	69.21	82.03	78.83	75.37	76.36	96.39
W4A4	OSFP4 W-SIC / X-RTN	69.73	**82.34**	**79.09**	76.95	**77.03**	**97.23**
Weights and activations: online activation-scale search							
W4A4	ScaleSweepMSE	68.99	82.18	78.92	**76.40**	76.62	96.72
W4A4	ScaleSweepMSE + GPTQ	69.53	82.94	78.51	75.77	76.69	96.80
W4A4	4over6	69.56	82.11	78.56	75.14	76.34	96.36
W4A4	OSFP4 W-RTN / X-ScaleSweepMSE	**69.72**	**83.55**	**79.36**	75.22	76.96	97.14
W4A4	OSFP4 W-SIC / X-ScaleSweepMSE	69.56	83.40	79.10	76.32	**77.09**	**97.31**
Weights only							
W4A16	RTN	70.69	83.85	79.36	76.80	77.67	98.04
W4A16	H-Scale	71.06	83.85	79.88	77.19	78.00	98.45
W4A16	OSFP4 W-RTN	71.13	83.47	79.77	**78.22**	78.15	98.64
W4A16	OSFP4 W-SIC	**71.53**	**84.15**	**80.01**	77.98	**78.42**	**98.98**

D.1 MODEL AND CALIBRATION

All locally prepared models start from the same BF16 Llama-3.1-8B-Instruct checkpoint and use the same calibration set: 1024 sequences of 2048 tokens sampled from FineWeb-Edu. All seven linear projections in each decoder layer are quantized. Embeddings, normalization parameters, the language-model head, and KV caches remain in BF16. NVFP4 uses E2M1 values with one E4M3 block scale per 16 input channels and an FP32 tensor scale. W4A4 also quantizes activations in groups of 16, whereas W4A16 retains BF16 activations. No fine-tuning or distillation is performed for the locally prepared models.

D.2 METHODS AND IMPLEMENTATION PROVENANCE

Author-provided implementations. The rows labeled RTN (FP-Quant), GPTQ (FP-Quant), and MR-GPTQ (FP-Quant) use the authors' quantization implementation (Egiazarian et al., 2026), run on our common checkpoint and calibration data. We convert the resulting models for native vLLM evaluation, preserving their quantized weights, scales, and transforms. These rows therefore use author-provided quantization with our serving integration, rather than the original paper's PyTorch evaluation backend. MR-GPTQ includes block Hadamard transforms and activation-based column ordering; GPTQ uses the original column order. All three use absmax activation scaling. SmoothQuant is omitted because the corresponding implementation from the FP-Quant authors was unavailable.

Our implementations of competing methods. The weight-only RTN control, ScaleSweepMSE, ScaleSweepMSE+GPTQ, 4over6, SOAR, and H-Scale rows use our own implementations of the respective procedures. ScaleSweepMSE (Lin & Wan, 2026) searches weight and activation block scales by reconstruction MSE; ScaleSweepMSE+GPTQ selects weight scales before the GPTQ feedback pass and holds them fixed during feedback. 4over6 (Cook et al., 2026) chooses between absmax-to-4 and absmax-to-6 scales for both weights and activations. SOAR (Bao et al., 2026) jointly refines tensor and block scales with separate encoding and decoding scales. H-Scale (Yu et al., 2026) selects weight scales using activation-weighted reconstruction error. SOAR and the H-Scale W4A4 extension use absmax activation scaling. H-Scale is originally a weight-only method; its W4A4 row is our extension, not a result reported by its authors. These independently implemented competitors use no rotations or column reordering.

OSFP4 and activation-scale ablations. The four OSFP4 configurations use the optimization and deterministic scale selection described in Appendices B and C. The diagonal transformation is folded into preceding normalization layers where possible and applied explicitly before the attention output and MLP down projections otherwise. The two activation-ScaleSweepMSE ablations reuse the corresponding OSFP4 W4A4 checkpoints, changing only the online activation block-scale rule. Their weights, diagonal transformations, and tensor scales remain fixed, so these rows isolate the effect of activation-scale selection.

Released NVIDIA checkpoint. We also evaluate the separately released `nvidia/Llama-3.1-8B-Instruct-NVFP4` checkpoint under the same task protocol. Its calibration and preparation are outside our controlled procedure, so it serves as a released-model reference rather than an isolated comparison of quantization algorithms.

The publishable code package contains the exact asset identities, calibration sampling procedure, software environments, method settings, conversion and validation procedures, and saved results needed to reproduce the table.

E Projection-Level Distortion and Deterministic Scale Selection

We compare the predicted randomized loss with measured projection errors for four configurations: W-RTN, W-SIC, W-RTN/X-RTN, and W-SIC/X-RTN. The first two retain unquantized activations (W4A16); the last two quantize both operands (W4A4). We use three fixed projections of Llama-3.1-8B-Instruct: the value projection in layer 12, the attention output projection in layer 26, and the MLP up-projection in layer 5, with zero-based layer indices. Each configuration and projection is optimized independently. The measurements concern local projection error on inputs captured from the BF16 model, rather than error propagated through an end-to-end quantized model.

Empirical activation distribution. We use the checkpoint and FineWeb-Edu calibration-token cache specified in Appendix D. From the 1024 cached document windows of length 2048, we select 128 distinct windows uniformly without replacement and 256 distinct token positions within each window. Causal BF16 forward passes, without a chat template or generated-token decoding, provide $N = 32768$ activation vectors $x_t \in \mathbb{R}^n$ for each projection. Here $n = 4096$ and the output dimensions are $m = 1024$, 4096, and 14336 for $12v$, $26o$, and $5\mathrm{up}$, respectively. The prompt/token pairs are identical across all configurations and projections. We regard x as uniform on this empirical population and use the same population for optimization and measurement; this is a calibration diagnostic, not a held-out generalization test.

Fitted parameters and comparisons. For each projection and configuration, we apply OSFP4 as described in Section 3 to obtain A, Γ_W and, for W4A4, Γ_X. We then keep these fitted parameters fixed and compare one predicted-loss reference with three measured error distributions.

Predicted loss (dashed line). For each configuration we plot

$$\overline{L} = \frac{L(A, \Gamma_W, \Gamma_X)}{Nm}, \tag{47}$$

where L is the corresponding loss from (22)–(25), with Γ_X omitted for W4A16. This prediction uses the optimized continuous scales, before deterministic E4M3 scale selection, and is the same reference for all curves in a panel.

Deterministic quantization (blue and orange). Writing $\widehat{W}$ for the reconstruction of WA^{-1} and $\widehat{x}_t$ for that of Ax_t, each histogram contains the per-activation output MSE

$$d_t = \frac{1}{m}\|\widehat{W}\widehat{x}_t - Wx_t\|_2^2. \tag{48}$$

The two curves test different scale-selection rules:

- **W4A16:** Blue uses OSFP4's deterministic weight-scale selection; orange replaces it with absmax-to-6 weight scales. Both retain the same fitted A, tensor scale, and RTN or SIC rounding procedure, with $\widehat{x}_t = Ax_t$. For SIC, both curves use Algorithm 1, with its scale-selection step replaced by absmax-to-6 for the orange curve.
- **W4A4:** Both deterministic curves use the same OSFP4-quantized weights, including the final E4M3 weight scales selected by the RTN or SIC procedure. Blue uses ScaleSweepMSE activation block scales, whereas orange uses absmax-to-6. The diagonal transformation A and all tensor scales remain fixed, so the comparison isolates activation block-scale selection.

Randomized pseudo-MSE (green). Using the fitted continuous scales, we apply the randomized quantizer to weights and, in W4A4, activations, retaining SIC feedback where applicable. To estimate the expectation over the random dither, we repeat this procedure 32 times with fresh independent dithers, sharing each randomized weight matrix across all activation samples. For each activation, we average the output MSE over these repetitions. The green histogram shows the distribution of these averages across activations.

The full W-RTN/W4A16 comparison is shown in Figure 4; its layer 12 value projection is shown separately in Figure 2 in the main text. The remaining configurations follow. Table 4 shows that the mean randomized pseudo-MSE closely matches the predicted loss across all configurations and projections, with ratios of 0.96–1.01. Deterministic OSFP4 achieves approximately half the predicted loss, with ratios of 0.48–0.53 (using ScaleSweepMSE activation scales in W4A4).

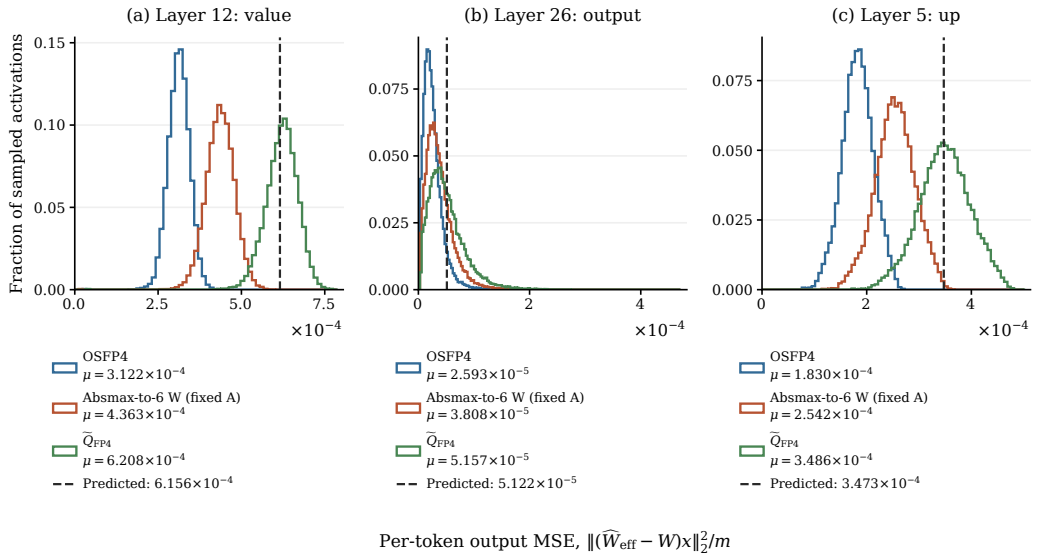


Figure 4: W-RTN (W4A16). Blue/orange: OSFP4/absmax-to-6 weight scales at fixed A. Green: randomized pseudo-MSE; dashed: predicted loss.

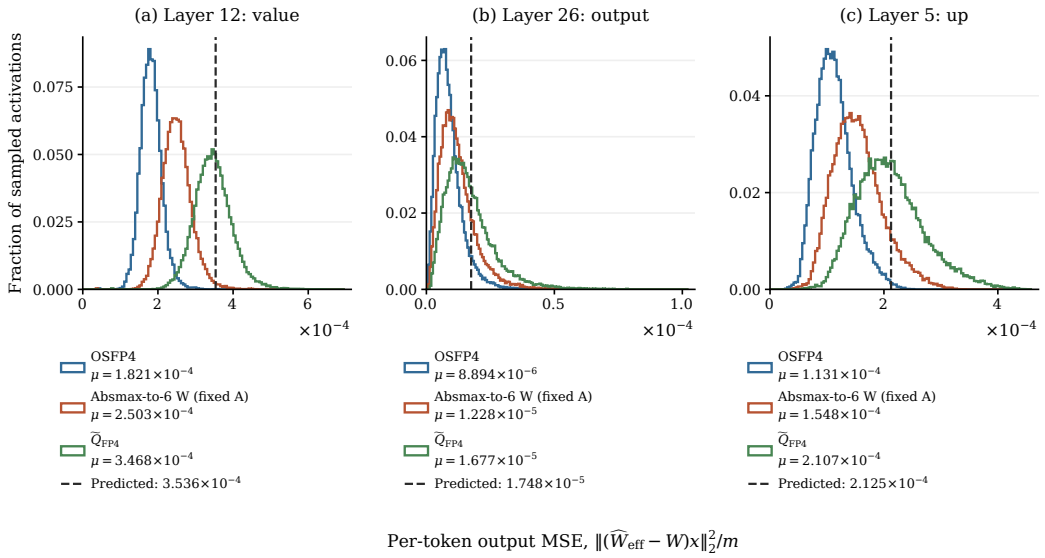


Figure 5: W-SIC (W4A16). Blue/orange: OSFP4/absmax-to-6 weight scales at fixed A. Green: randomized pseudo-MSE; dashed: predicted loss.

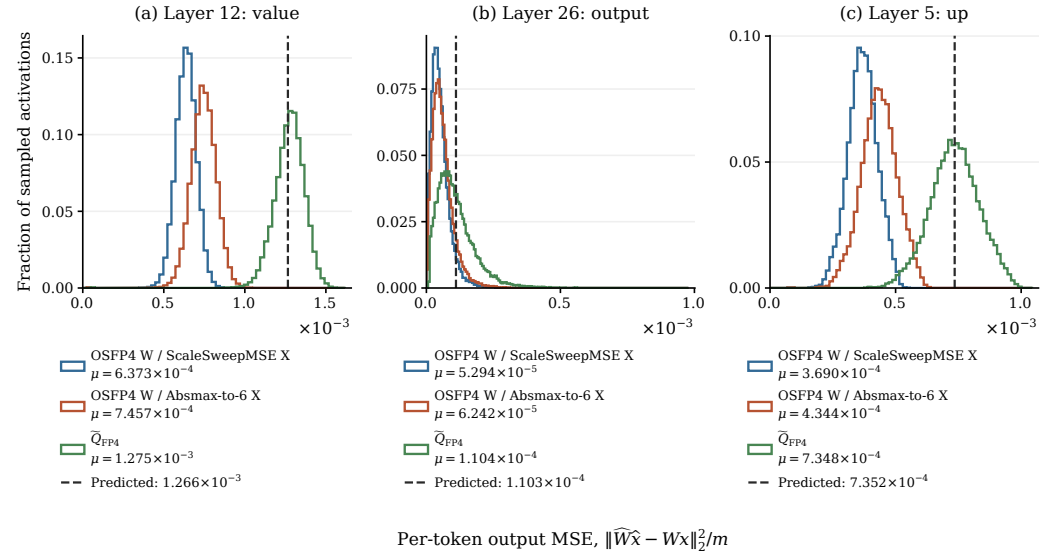


Figure 6: W-RTN/X-RTN (W4A4). Blue/orange: ScaleSweepMSE/absmax-to-6 activation scales with identical OSFP4 weights. Green: randomized pseudo-MSE; dashed: predicted loss.

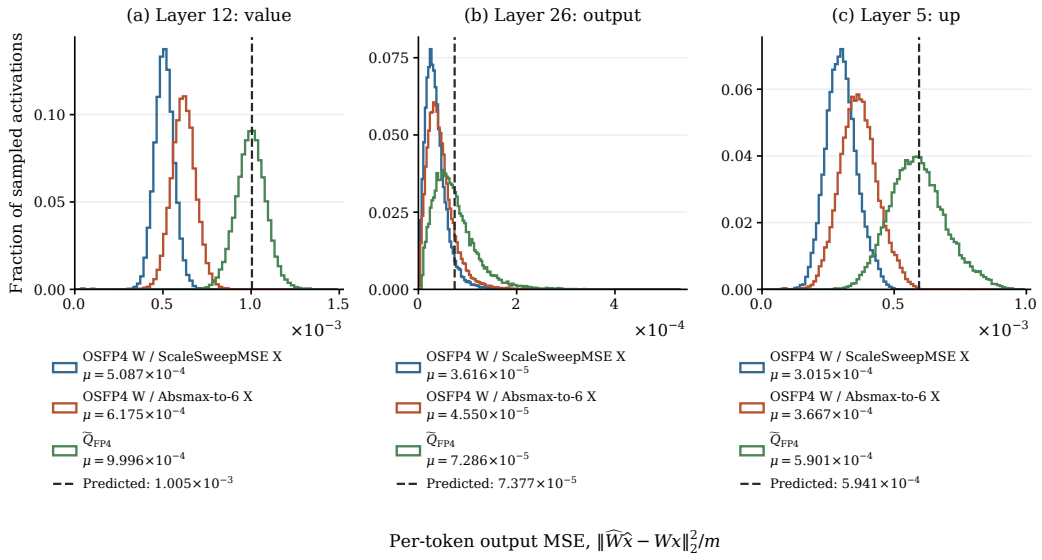


Figure 7: W-SIC/X-RTN (W4A4). Blue/orange: ScaleSweepMSE/absmax-to-6 activation scales with identical OSFP4 weights. Green: randomized pseudo-MSE; dashed: predicted loss.

Table 4: Mean measured MSE relative to predicted loss. OSFP4 W4A4 uses ScaleSweepMSE activation scales.

	Randomized mean / prediction			**OSFP4 mean / prediction**		
Configuration	12v	26o	5up	12v	26o	5up
W-RTN	1.0085	1.0068	1.0038	0.5072	0.5062	0.5267
W-SIC	0.9807	0.9595	0.9913	0.5149	0.5088	0.5320
W-RTN/X-RTN	1.0075	1.0008	0.9994	0.5035	0.4798	0.5019
W-SIC/X-RTN	0.9950	0.9876	0.9932	0.5063	0.4902	0.5076

F ADDITIONAL MODELS

This appendix reports additional model results.

F.1 LLAMA-3-8B PERPLEXITY

Published comparison results are taken from Table 2 of Savkin et al. (2025). The OSFP4, GPTQ and RTN results are added here.

Table 5: WikiText-2 perplexity comparison for Llama-3-8B.

Precision	**Method**	**Llama-3-8B**
Baseline	BF16	6.14
W4A16	NestQuant	**6.31**
	NestQuantM	6.35
	RTN	6.54
	GPTQ	6.37
	W-RTN	6.43
	W-SIC	**6.31**
W4A4	SpinQuant	7.10
	OstQuant	7.24
	DuQuant	8.06
	RTN	6.88
	GPTQ	6.68
	W-RTN/X-RTN	6.74
	W-SIC/X-RTN	**6.59**
W4A4KV4	NestQuant	**6.63**
	NestQuantM	6.82

F.2 QWEN3-8B ACCURACY

Table 6: Evaluation comparison on Qwen3-8B. All quantized configurations use the NVFP4 format.

Precision	**Method**	**MMLU-CoT**	**GSM8K**	**HellaSwag**	**WinoGrande**	**Avg.**	**Recovery %**
Baseline	BF16	71.64	90.98	76.60	70.64	**77.46**	**100.00**
Weights and activations							
W4A4	RTN (FP-Quant)	67.72	89.31	74.68	69.46	75.29	97.20
W4A4	GPTQ (FP-Quant)	70.94	89.92	74.63	69.61	76.28	98.48
W4A4	MR-GPTQ (FP-Quant)	**73.60**	89.76	74.92	68.43	76.68	98.99
W4A4	QAT+Had16 (FP-Quant)	71.29	89.16	74.63	**70.32**	76.35	98.57
W4A4	OSFP4 W-RTN / X-RTN	73.22	**90.14**	74.69	69.30	**76.84**	**99.20**
W4A4	OSFP4 W-SIC / X-RTN	70.49	89.61	**75.24**	70.01	76.34	98.55
Weights only							
W4A16	OSFP4 W-RTN	**72.28**	**90.75**	75.76	**70.96**	**77.44**	**99.97**
W4A16	OSFP4 W-SIC	69.87	90.07	**76.05**	70.09	76.52	98.79

F.3 QWEN3-30B-A3B-INSTRUCT ACCURACY

For each MoE layer, we use expert-specific diagonal smoothing matrices rather than sharing a single transformation across experts. During calibration, we preserve the model's natural MoE routing, so each expert is calibrated only on the tokens routed to it. At runtime, after routing, $A_{\text{gate/up}}$ is applied to the shared input activations of the selected expert's gate and up projections, while A_{down} is applied to the input activations of its down projection.

For OSFP4 W4A4, activation global scales would need to be computed from smoothed calibration activations separately for each expert's shared gate/up input and down input. Inference would then require a vLLM plugin that selects the appropriate scale after routing. Because this route-aware serving path was not implemented in our experiments, we report only W4A16 results for this model.

For this evaluation, we measure every row once using the same seven-task protocol. We evaluate C-Eval, LiveBench, MMLU-Redux, AIME 2025, ARC-Challenge, BBH, and GPQA Diamond. Generation uses temperature 0.7, top-p 0.8, top-k 20, repetition penalty 1.0, presence penalty 1.5, and at most 32,768 output tokens. AIME 2025 uses 16 independent generations per problem. We report the unweighted mean of the seven task scores.

Table 7: Evaluation comparison on Qwen3-30B-A3B-Instruct.

Precision	Method	C-Eval	LiveBench	MMLU-R	AIME 2025	ARC-C	BBH	GPQA-D	Avg.	Recovery %
Baseline	BF16	86.97	70.41	88.19	63.96	95.14	90.77	55.05	**78.64**	**100.00**
Weights only										
W4A16	RTN	86.57	67.11	87.63	57.71	**95.56**	89.42	**61.11**	77.87	99.02
W4A16	RTN + H-Scale	86.37	68.22	87.67	**61.46**	**95.56**	89.54	57.07	77.98	99.16
W4A16	GPTQ + H-Scale	86.22	67.93	87.72	59.58	95.22	90.06	55.56	77.47	98.51
W4A16	MR-GPTQ + H-Scale	86.51	67.24	**87.82**	60.42	95.14	89.71	54.04	77.27	98.26
W4A16	OSFP4 W-RTN	**86.70**	**69.44**	87.61	60.21	95.14	**90.08**	58.08	**78.18**	**99.42**
W4A16	OSFP4 W-SIC	86.39	67.77	87.77	57.92	94.71	89.96	57.58	77.44	98.47

F.4 GEMMA-4-31B-IT ACCURACY

For this evaluation, we use the no-thinking setting and seed counts described in Red Hat AI's Gemma-4-31B-IT-NVFP4 model card on Hugging Face as a reference for our local measurements. Every row uses zero-shot, text-only prompts and the Gemma chat template with thinking disabled. We generate responses with vLLM using a 32,768-token context, temperature 1.0, top-p 0.95, and top-k 64. We score GPQA Diamond (GPQA-D), MATH-500, and AIME 2025 with LightEval, and IFEval with lm-eval's prompt-level strict metric. Scores are averaged over three seeds for GPQA-D, MATH-500, and IFEval, and eight seeds for AIME 2025. The RTN (RedHatAI) row uses RedHatAI's publicly released NVFP4 checkpoint. All table entries, including BF16, are our local measurements.

Table 8: Evaluation comparison on Gemma-4-31B-IT.

Precision	Method	GPQA-D	MATH-500	IFEval strict	AIME 2025	Avg.	Recovery %
Baseline	BF16	78.28	89.20	90.94	67.92	81.59	**100.00**
Weights and activations							
W4A4	RTN (RedHatAI)	76.60	88.07	90.14	**64.17**	79.74	97.74
W4A4	MR-GPTQ (FP-Quant)	74.58	88.20	**91.00**	63.75	79.38	97.30
W4A4	OSFP4 W-RTN / X-RTN	75.76	90.40	90.51	63.75	80.11	98.19
W4A4	OSFP4 W-SIC / X-RTN	**78.28**	**91.87**	90.76	62.08	**80.75**	**98.97**

G INFERENCE EFFICIENCY

Table 9 compares the inference throughput of OSFP4, the vendor NVFP4 RTN baseline, NVFP4 with unit smoothing, and BF16 on Llama-3.1-8B-Instruct (NVIDIA B200, vLLM 0.28.0). Each workload uses a fixed synthetic batch of random-token prompts. Prefill emits one token and measures time to first token. Decode is measured with 256 output tokens and with one output token; subtracting the two durations removes the common prefill contribution. For each method and workload, we perform three independent timing repetitions, each consisting of 30 inference runs. The smoothing transformation A is applied at runtime only on `o_proj` and `down_proj`.

End-to-end inference includes attention with a BF16 KV cache, activation quantization, output writes, and scheduling overhead in addition to FP4 matrix multiplication. The measured gains therefore depend on the workload. OSFP4 closely matches the unit-smoothing control across the sweep, suggesting that its gap to the vendor NVFP4 path primarily reflects the runtime smoothing implementation rather than the values of A.

Table 9: Inference throughput on Llama-3.1-8B-Instruct (NVIDIA B200).

Workload			**Throughput**			
Batch	**Prompt**	**Out**	**BF16**	**NVFP4**	**NVFP4** $A{=}I$	**OSFP4**
Prefill: input throughput (thousand tokens/s), measured to first output token						
1	512	1	60.2	81.0	78.1	78.4
8	512	1	82.5	177.0	165.4	166.5
32	512	1	86.5	192.4	180.8	180.6
1	2048	1	77.0	153.1	145.0	144.9
8	2048	1	85.0	183.9	173.7	174.0
1	8192	1	75.0	144.6	137.8	138.0
Decode: output throughput across the batch (tokens/s), excluding prefill						
1	128	256	274	321	312	312
8	128	256	2183	2627	2548	2547
32	128	256	8448	9936	9664	9642

H COMPONENT ABLATION

We conduct an ablation study on Llama-3.1-8B-Instruct under W4A16 and W4A4 quantization. All configurations are calibrated on $1{,}024$ FineWeb sequences of $2{,}048$ tokens. Perplexity and projection-output errors are evaluated on the WikiText-2 test set.

For W4A16, we compare *AbsMax*, *optimal E4M3 scale according to unweighted MSE*, *our W-RTN method*, *W-SIC with $A = I$ and standard AbsMax weight scales*, and *our W-SIC method*.

For W4A4, we compare *AbsMax*, *optimal E4M3 scale for W, with $A = I$ and standard AbsMax scaling for X*, *our W-RTN method*, *W-SIC with $A = I$ and standard AbsMax weight and activation scales*, and *our W-SIC method*.

For each method, we report the relative MatMul MSE for the seven projection types $p \in \{q, k, v, o, \mathrm{gate}, \mathrm{up}, \mathrm{down}\}$. Let $Y_{t,p}$ denote the full-precision output of projection p in Transformer layer t, and let $\hat{Y}_{t,p}^{\mathrm{method}}$ and $\hat{Y}_{t,p}^{\mathrm{FP8\ RTN}}$ denote the corresponding outputs produced by the evaluated method and the FP8 RTN reference, respectively. We compute the squared Frobenius-error ratio independently in each Transformer layer and report its mean over all 32 layers:

$$R_p = \frac{1}{32} \sum_{t=1}^{32} \frac{\left\| \hat{Y}_{t,p}^{\mathrm{method}} - Y_{t,p} \right\|_F^2}{\left\| \hat{Y}_{t,p}^{\mathrm{FP8\ RTN}} - Y_{t,p} \right\|_F^2}, \quad \text{for } p \in \{q, k, v, o, \mathrm{gate}, \mathrm{up}, \mathrm{down}\}. \tag{49}$$

The W4A16 configurations use a W8A16 FP8 RTN reference, whereas the W4A4 configurations use a W8A8 FP8 RTN reference. The table reports the seven relative MatMul MSE values and WikiText-2 perplexity for each configuration.

Table 10: NVFP4 component ablation on Llama-3.1-8B-Instruct: relative projection MSE and WikiText-2 perplexity.

Precision	**Method**	**Relative MMSE ↓**							**PPL ↓**
		q	***k***	***v***	***o***	**gate**	**up**	**down**	
Baseline	BF16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**7.22**
Weights only									
W8A16	RTN	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**7.24**
W4A16	RTN	10.27	8.68	14.30	12.59	12.22	12.64	12.52	7.55
W4A16	MSE	8.12	6.93	12.01	9.28	9.08	9.39	9.43	7.52
W4A16	W-RTN	5.29	4.42	8.53	8.16	8.48	8.95	8.10	7.42
W4A16	SIC	5.88	4.73	9.05	5.82	9.00	9.53	9.19	7.45
W4A16	W-SIC	**3.63**	**2.89**	**6.08**	**4.02**	**6.45**	**6.86**	**6.29**	**7.39**
Weights and activations									
W8A8	RTN	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**7.27**
W4A4	RTN	10.86	8.97	16.18	12.67	12.41	12.85	12.45	7.88
W4A4	MSE	9.76	8.09	15.00	10.96	10.80	11.20	10.81	7.85
W4A4	W-RTN/X-RTN	7.05	6.19	10.12	10.12	10.38	10.75	9.77	7.77
W4A4	SIC	8.52	6.88	13.54	9.25	10.75	11.28	10.68	7.77
W4A4	W-SIC/X-RTN	**6.17**	**5.38**	**8.94**	**8.03**	**9.33**	**9.72**	**8.82**	**7.67**

I Detailed Loss Derivations and SIC Procedure

Denote by $\mathcal{C}_{\text{scale}}$ all the positive points in the FP8 E4M3 constellation, and recall the definition of $\mathcal{C}_{\text{FP4}}$ from (1).

Let $W \in \mathbb{R}^{m\times n}$ and $X \in \mathbb{R}^{n\times k}$. We generally think of X as a random vector with $k = 1$, but for optimization of the diagonal scaling matrix A, we will denote by X the matrix holding k calibration activations in $\mathbb{R}^n$, so that $k > 1$. We also define $\Sigma_X = \frac{1}{k}XX^\top \in \mathbb{R}^{n\times n}$. Let $L = n/16$ and assume for simplicity that L is an integer.

We partition W to L matrices of shape $m \times 16$ so that

$$W = [W_1|\cdots|W_L], \quad W_\ell = W(:, 1 + 16(\ell-1) : 16\ell) \text{ for } \ell = 1, \ldots, L. \tag{50}$$

Under NVFP4 quantization we represent each W_ℓ as

$$\hat{W}_\ell = \text{diag}(\Gamma_{W_\ell}) \cdot \hat{W}_\ell^{\text{FP4}}, \text{ where } \Gamma_{W_\ell} \in \mathcal{C}_{\text{scale}}^m, \hat{W}_\ell^{\text{FP4}} \in \mathcal{C}_{\text{FP4}}^{m\times 16}. \tag{51}$$

Similarly,

$$X = \begin{bmatrix} \frac{X_1}{\vdots} \\ \hline X_L \end{bmatrix}, \quad X_\ell = X(1 + 16(\ell-1) : 16\ell, :) \text{ for } \ell = 1, \ldots, L. \tag{52}$$

When activations are also quantized, we represent each X_ℓ as

$$\hat{X}_\ell = \hat{X}_\ell^{\text{FP4}} \cdot \text{diag}(\Gamma_{X_\ell}), \text{ where } \Gamma_{X_\ell} \in \mathcal{C}_{\text{scale}}^k, \hat{X}_\ell^{\text{FP4}} \in \mathcal{C}_{\text{FP4}}^{16\times k}. \tag{53}$$

In this work we use a diagonal smoothing matrix $A \in \mathbb{R}^{n\times n}$ so that we actually quantize WA^{-1} and (possibly) AX. We keep using the notation $\{\hat{W}_\ell\}$ and $\{\hat{X}_\ell\}$ for the corresponding NVFP4 representation even when $A \neq I_n$, in which case $\hat{W}_\ell$ can be very far from W_ℓ, and same for X. We also express the matrix A as a block diagonal matrix with blocks of size 16×16 so that

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_L \end{bmatrix}. \tag{54}$$

The MatMul Quantization Error: Let

$$e = \hat{W}\hat{X} - WX = \sum_{\ell=1}^{L} \left(\hat{W}_\ell \hat{X}_\ell - (W_\ell A_\ell^{-1})(A_\ell X_\ell) \right) \tag{55}$$

$$= \sum_{\ell=1}^{L} (W_\ell A_\ell^{-1} + e_{W_\ell})(A_\ell X_\ell + e_{X_\ell}) - \left((W_\ell A_\ell^{-1})(A_\ell X_\ell) \right) \tag{56}$$

$$= \sum_{\ell=1}^{L} e_{W_\ell} A_\ell X_\ell + W_\ell A_\ell^{-1} e_{X_\ell} + e_{W_\ell} e_{X_\ell} \tag{57}$$

where $e_{W_\ell} = \hat{W}_\ell - (W_\ell A_\ell^{-1})$ and $e_{X_\ell} = \hat{X}_\ell - (A_\ell X_\ell)$. Setting

$$e_W = [e_{W_1}|\cdots|e_{W_L}], \quad e_X = \begin{bmatrix} \frac{e_{X_1}}{\vdots} \\ \hline e_{X_L} \end{bmatrix}, \tag{58}$$

we obtain

$$e = e_W AX + WA^{-1} e_X + e_W e_X. \tag{59}$$

We further neglect the third term which has much smaller energy than the sum of the first two under NVFP4 quantization, and approximate

$$e \approx e_W AX + WA^{-1} e_X. \tag{60}$$

Randomized FP4 quantization error: We assume the randomized quantizer $\tilde{Q}_{\mathrm{FP4}}(\cdot)$ from (3) is used. We model it as a channel with input $y \in \mathbb{R}$ and output $\tilde{Q}_{\mathrm{FP4}}(y) \in \mathbb{R}$ satisfying

$$\tilde{Q}_{\mathrm{FP4}}(y) = y + \sqrt{y^2\phi(y)}Z, \tag{61}$$

where Z is a random variable with zero mean and unit variance, statistically independent of everything. For a matrix $F \in \mathbb{R}^{k_1 \times k_2}$ the notation $\tilde{Q}_{\mathrm{FP4}}(F)$ corresponds to applying the randomized quantizer $\tilde{Q}_{\mathrm{FP4}}(\cdot)$ on each component of F.

Round to nearest (RTN) quantization: The simplest way to use the randomized quantizer is to feed to it the scaled signal, and multiply the output by the inverse scaling. This is referred to as RTN quantization. Specifically, given A, Γ_W under RTN quantization the entries of the weight matrix W are quantized to

$$\hat{W}_\ell^{\mathrm{FP4}} = \tilde{Q}_{\mathrm{FP4}}\left(\mathrm{diag}(\Gamma_{W_\ell})^{-1} W_\ell A_\ell^{-1}\right), \quad \text{so that } \hat{W}_\ell = \mathrm{diag}(\Gamma_{W_\ell})\tilde{Q}_{\mathrm{FP4}}\left(\mathrm{diag}(\Gamma_{W_\ell})^{-1} W_\ell A_\ell^{-1}\right). \tag{62}$$

Denoting $\{\gamma_{W_\ell}^{(i)}\}_{i=1}^m$ and $\{\alpha_\ell^{(r)}\}_{r=1}^{16}$ for the diagonal entries of Γ_{W_ℓ} and A_ℓ, respectively, we obtain

$$e_{W_\ell}(i,r) = \hat{W}_\ell(i,r) - \frac{W_\ell(i,r)}{\alpha_\ell^{(r)}} = \sqrt{\left(\frac{W_\ell(i,r)}{\alpha_\ell^{(r)}}\right)^2 \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right)} Z_\ell(i,r), \tag{63}$$

where $\{Z_\ell(i,r)\}$ are iid zero-mean, unit-variance random variables.

Let

$$L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) = \mathbb{E}\|e_{W_\ell} A_\ell X_\ell\|_F^2, \tag{64}$$

where $X_\ell, W_\ell, \Gamma_{W_\ell}, A_\ell$ are assumed fixed and the expectation is only with respect to the randomization of $\tilde{Q}_{\mathrm{FP4}}(\cdot)$, that is, with respect to the iid random variables $\{Z_\ell(i,r)\}$. It immediately follows that

$$\begin{aligned} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) &= \sum_{i=1}^{m}\sum_{j=1}^{k}\sum_{r=1}^{16} W_\ell^2(i,r) X_\ell^2(r,j) \cdot \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right) \\ &= k\sum_{r=1}^{16}\sigma_{X_\ell}^2(r)\sum_{i=1}^{m} W_\ell^2(i,r)\cdot\phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)}\alpha_\ell^{(r)}}\right), \end{aligned} \tag{65}$$

where $\sigma_{X_\ell}^2(r) = \frac{1}{k}\sum_{j=1}^k X_\ell^2(r,j)$ (we also have $\sigma_{X_\ell}^2(r) = \Sigma_X(r+16(\ell-1), r+16(\ell-1))$).

We assume that the random variables $\{Z_\ell(\cdot,\cdot)\}$ corresponding to different blocks are also independent. This implies

$$L_W^{\mathrm{RTN}} = \mathbb{E}\|e_W A X\|_F^2 = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell). \tag{66}$$

If activations are also quantized, applying RTN with the randomized FP4 quantizer gives

$$\hat{X}_\ell^{\mathrm{FP4}} = \tilde{Q}_{\mathrm{FP4}}\left(A_\ell X_\ell \,\mathrm{diag}(\Gamma_{X_\ell})^{-1}\right), \quad \text{so that } \hat{X}_\ell = \tilde{Q}_{\mathrm{FP4}}\left(A_\ell X_\ell \,\mathrm{diag}(\Gamma_{X_\ell})^{-1}\right)\mathrm{diag}(\Gamma_{X_\ell}), \tag{67}$$

and

$$e_{X_\ell}(r,j) = \hat{X}_\ell(r,j) - \alpha_\ell^{(r)} X_\ell(r,j) = \sqrt{\left(\alpha_\ell^{(r)} X_\ell(r,j)\right)^2 \phi\left(\frac{\alpha_\ell^{(r)}}{\gamma_{X_\ell}^{(j)}} X_\ell(r,j)\right)}\tilde{Z}_\ell(r,j), \tag{68}$$

where $\{\gamma_{X_\ell}^{(j)}\}_{j=1}^k$ denote the diagonal entries of Γ_{X_ℓ}, and $\{\tilde{Z}_\ell(r,j)\}$ are iid zero-mean unit-variance random variables statistically independent of everything. Similar to the derivation above, we obtain

$$L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell) = \mathbb{E}\|W_\ell A_\ell^{-1} e_{X_\ell}\|_F^2 = m \sum_{r=1}^{16} \sigma_{W_\ell}^2(r) \sum_{j=1}^{k} X_\ell^2(r,j) \cdot \phi\left(\frac{\alpha_\ell^{(r)}}{\gamma_{X_\ell}^{(j)}} X_\ell(r,j)\right), \quad (69)$$

where $\sigma_{W_\ell}^2(r) = \frac{1}{m}\sum_{i=1}^{m} W_\ell^2(i,r)$. We also have

$$L_X^{\mathrm{RTN}} = \mathbb{E}\|WA^{-1}e_X\|_F^2 = \sum_{\ell=1}^{L} L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell). \quad (70)$$

Successive interference cancellation (SIC): Note that

$$\|e_W AX\|_F^2 = \mathrm{tr}\left((e_W AX)(e_W AX)^\top\right) = k\,\mathrm{tr}\left(e_W A \Sigma_X A^\top e_W^\top\right). \quad (71)$$

Let $U \in \mathbb{R}^{n\times n}$ be an upper triangular matrix such that $\Sigma_X = U^\top U$, and denote

$$Y = UA^\top (WA^{-1})^\top = UW^\top \in \mathbb{R}^{n\times m}. \quad (72)$$

We have

$$\|e_W AX\|_F^2 = k\|UA^\top e_W^\top\|_F^2 = k\|Y - UA\hat{W}^\top\|_F^2. \quad (73)$$

Even when A and the E4M3 scaling matrices $\{\Gamma_{W_\ell}\}_{\ell=1}^{L}$ are fixed, the problem of choosing $\hat{W}^{\mathrm{FP4}} \in \mathcal{C}_{\mathrm{FP4}}^{m\times n}$ such as to minimize (73) is computationally difficult due to the coupling the matrix UA introduces between different columns of $\hat{W}$ (equivalently, rows of $\hat{W}^\top$). As common in the literature, instead of attempting an optimal solution, we resort to the SIC/GPTQ approach. Taking into account the structure of the NVFP4 constellation, this boils down to choosing $W_{\mathrm{SIC}} \in \mathcal{C}_{\mathrm{FP4}}^{n\times m}$ as in Algorithm 2, and then setting $\hat{W}^{\mathrm{FP4}} = W_{\mathrm{SIC}}^\top$ and $\hat{W}_\ell = \Gamma_{W_\ell}\hat{W}_\ell^{\mathrm{FP4}}$.

Algorithm 2 NVFP4 SIC

Inputs: $Y \in \mathbb{R}^{n\times m}$, upper triangular $U \in \mathbb{R}^{n\times n}$, diagonal matrices $\{A_\ell\}_{\ell=1}^{L=n/16}$ with $A_\ell = \mathrm{diag}\left(\alpha_\ell^{(1)}, \ldots, \alpha_\ell^{(16)}\right)$, and E4M3 diagonal scaling matrices $\{\Gamma_{W_\ell}\}_{\ell=1}^{L=n/16}$ with $\Gamma_{W_\ell} = \mathrm{diag}\left(\gamma_{W_\ell}^{(1)}, \ldots, \gamma_{W_\ell}^{(m)}\right)$

Outputs: $W_{\mathrm{SIC}} \in \mathcal{C}_{\mathrm{FP4}}^{n\times m}$

$Y^{(n)} \leftarrow Y$
for $\ell = L : 1$ **do**
  for $r = 16 : 1$ **do**
    $t \leftarrow 16(\ell-1) + r$
    $W_{\mathrm{SIC}}(t,:) \leftarrow Q_{\mathrm{FP4}}\left(\frac{Y^{(t)}(t,:)\Gamma_{W_\ell}^{-1}}{\alpha_\ell^{(r)} U(t,t)}\right)$ ▷ For optimization use $\tilde{Q}_{\mathrm{FP4}}$, for actual quantization use Q_{FP4}
    $Y^{(t-1)} \leftarrow Y^{(t)} - \alpha_\ell^{(r)} U(:,t) \cdot W_{\mathrm{SIC}}(t,:)\Gamma_{W_\ell}$ ▷ $U(:,t)$ is the tth column of U
  end for
end for

The analysis of $L_W^{\mathrm{SIC}} = k\mathbb{E}\|Y - UA\hat{W}^\top\|_F^2$ under the SIC algorithm is a bit complicated, even under the randomized FP4 quantization error model (61). The reason is that Z entries from time t affect $Y^{(t')}$ for $t' < t$ due to the feedback nature of the quantization process. We therefore make a high-resolution quantization assumption and assume the energy of the (filtered) quantization noise is negligible next to the signal (entries of Y) energy. Under this assumption we have that

$$Y^{(t)}(t,:) \approx U(t,t) \cdot (W(:,t))^\top. \quad (74)$$

With this assumption a straightforward calculation gives (under the randomized FP4 quantization error model (61)) for fixed $\{A_\ell\}$ and $\{\Gamma_{W_\ell}\}$

$$L_W^{\mathrm{SIC}} = k \sum_{\ell=1}^{L} \sum_{r=1}^{16} U_\ell^2(r) \sum_{i=1}^{m} W_\ell^2(i,r) \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)} \alpha_\ell^{(r)}}\right) = \sum_{\ell=1}^{L} L_W^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell), \quad (75)$$

where $U_\ell(r) = U(16(\ell-1)+r, 16(\ell-1)+r)$ and

$$L_W^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) = k \sum_{r=1}^{16} U_\ell^2(r) \sum_{i=1}^{m} W_\ell^2(i,r) \phi\left(\frac{W_\ell(i,r)}{\gamma_{W_\ell}^{(i)} \alpha_\ell^{(r)}}\right). \tag{76}$$

Note that the only difference between $L_W^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell)$ and $L_W^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell)$ is replacing $\sigma_{X_\ell}^2(r)$ in the latter with $U_\ell^2(r)$ in the former. Since $U_\ell^2(r)$ is the variance of the optimal prediction error of $X(t)$ - the tth coordinate of $X \sim \mathcal{N}(0, \Sigma_X)$, $t = 16(\ell-1)+r$ - from $X(t+1:n)$ and $\sigma_{X_\ell}^2(r)$ is the variance of $X(t)$, we have $U_\ell^2(r) \le \sigma_{X_\ell}^2(r)$ for all ℓ, r.

Total quadratic Distortion: Using (65) (69) and (76) we obtain the following loss functions for randomized NVFP4 quantization, as a function of A and the E4M3 scales, under the various quantization setups:

$$L^{W-\mathrm{RTN}} = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) \tag{77}$$

$$L^{W-\mathrm{SIC}} = \sum_{\ell=1}^{L} L_{W_\ell}^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) \tag{78}$$

$$L^{W-\mathrm{RTN},X-\mathrm{RTN}} = \sum_{\ell=1}^{L} \left[L_{W_\ell}^{\mathrm{RTN}}(\Gamma_{W_\ell}, A_\ell) + L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell)\right] \tag{79}$$

$$L^{W-\mathrm{SIC},X-\mathrm{RTN}} = \sum_{\ell=1}^{L} \left[L_{W_\ell}^{\mathrm{SIC}}(\Gamma_{W_\ell}, A_\ell) + L_{X_\ell}^{\mathrm{RTN}}(\Gamma_{X_\ell}, A_\ell)\right]. \tag{80}$$

Note that in all expressions the dependency on A and the E4M3 scales *completely decouples to L different functions, that depend on different parameters*. Therefore, the optimization on $\{A_\ell, \Gamma_{W_\ell}, \Gamma_{X_\ell}\}$ decouples to L independent optimization problems.