INT V.S. FP: A COMPREHENSIVE STUDY OF FINE-GRAINED LOW-BIT QUANTIZATION FORMATS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern AI hardware, such as Nvidia's Blackwell architecture, is increasingly embracing low-precision floating-point (FP) formats to handle the pervasive activation outliers in Large Language Models (LLMs). Despite this industry trend, a unified comparison of FP and integer (INT) quantization across varying granularities has been missing, leaving algorithm and hardware co-design without clear guidance. This paper fills that gap by systematically investigating the trade-offs between FP and INT formats. We reveal a critical performance crossover: while FP excels in coarse-grained quantization, INT consistently surpasses it as the quantization block size shrinks. Our comprehensive comparison demonstrates that for popular fine-grained formats like MX (block size 32), MXINT8 and MXINT4 are superior to their FP counterparts in both algorithmic accuracy and hardware efficiency. We also introduce a symmetric clipping method that resolves gradient bias in fine-grained low-bit INT training, enabling nearly lossless performance for MXINT8 training. These findings challenge the current hardware trajectory and advocate for prioritizing fine-grained INT formats in future AI accelerators to achieve a better balance of accuracy, power, and efficiency.

1 INTRODUCTION

The proliferation of Large Language Models (LLMs) has been accompanied by a surge in their computational and memory demands (Yuan et al., 2024), making quantization an indispensable technique for efficient deployment. A central challenge in quantizing LLMs, particularly those based on the Transformer architecture, is the presence of significant outliers (Sun et al., 2024; Dettmers et al., 2022) in activation distributions. These outliers, characterized by their large magnitude but infrequent occurrence, pose a considerable problem for low-precision representations. To accommodate this wide dynamic range, the AI hardware industry (NVIDIA Corporation, 2024a) is increasingly pivoting towards low-precision floating-point (FP) formats, such as FP8 and FP4. Prominent examples like NVIDIA's Blackwell architecture (NVIDIA Corporation, 2024a) underscore this trend, favoring the superior dynamic range of FP to handle outliers more gracefully than traditional integer (INT) formats.

However, this industry-wide momentum towards FP formats is based on an incomplete picture. The comparative advantages of FP and INT have not been systematically evaluated across different quantization granularities in a unified framework. Most studies (Xiao et al., 2023; Chen et al., 2024a; Liu et al., 2024b) focus on a single format or compare them only at coarse granularities (e.g., per-channel), failing to answer a critical question: how does the performance trade-off between INT and FP evolve as granularity becomes finer? Since fine-grained (block-wise) quantization is now a standard technique (Rouhani et al., 2023; NVIDIA Corporation, 2024b) for mitigating outliers, understanding its interaction with the underlying number format is essential for effective algorithm-hardware co-design.

In this paper, we conduct a comprehensive, systematic comparison of INT and FP quantization across a wide spectrum of block sizes. Our investigation reveals a critical "crossover point" in performance. While FP formats hold a distinct advantage in coarse-grained scenarios, we find that INT formats consistently surpass them as the block size shrinks. This reversal occurs because fine-grained blocking effectively isolates outliers, reducing the local dynamic range within each block and allowing the uniform precision of INT formats to become more effective. This trend holds across

modern block-wise formats, where a shared scaling factor is applied to a group of values, such as the 32-element blocks in Microscaling (MX) formats (Rouhani et al., 2023) or the 16-element blocks in NVIDIA's (NV) formats (NVIDIA Corporation, 2024b). To enable a direct comparison, we introduce and evaluate integer variants (e.g., MXINT8, MXINT6, MXINT4, NVINT4) alongside their standard FP counterparts (e.g., MXFP8, MXFP6, MXFP4, NVFP4). Our key contributions are as follows:

- We provide the first comprehensive comparison of INT versus FP quantization across a spectrum of granularities, demonstrating that fine-grained integer formats (MXINT8, MXINT4, and NVINT4) consistently outperform their FP counterparts in both direct-cast inference and low-bit training scenarios.
- We identify and resolve a critical gradient bias issue in fine-grained INT quantization-aware training (QAT) by introducing symmetric clipping method, enabling MXINT8 training to match the performance of BF16 training.
- We develop a theoretical and statistical framework that models the quantization error for both INT and FP formats, clearly explaining why INT surpasses FP in fine-grained regimes by analyzing the impact of the crest factor on quantization signal-to-noise ratio (QSNR).
- We present a comparative hardware cost analysis, showing that fine-grained INT quantization is not only more accurate but also more area- and energy-efficient than its FP equivalent.
- Collectively, our findings challenge the prevailing FP-centric trajectory in AI hardware design and strongly suggest that fine-grained INT formats offer a more optimal balance of accuracy and efficiency for the next generation of LLMs.

2 PRELIMINARIES

Quantization maps a high-precision tensor $\mathbf{X}$ to a lower bit-width. In this section, we present low-bit integer (INT) quantization, floating-point (FP) quantization, quantization granularity with a focus on fine-grained block-wise schemes, and an overview of existing low-bit block formats.

2.1 LOW-PRECISION INTEGER FORMATS

For b-bit integer quantization, we define:

$$\mathbf{X_q} = \text{clip}\left(\left\lfloor \frac{\mathbf{X}}{s} \right\rceil, Q_{\min}, Q_{\max}\right) \cdot s, \tag{1}$$

where s is the scale factor that normalizes $\mathbf{X}$ to the target integer range, $\lfloor \cdot \rceil$ is round-to-nearest, and $\mathbf{X_q}$ is the dequantized tensor. The clipping ensures that the integer values lie in $[Q_{\min}, Q_{\max}]$ (e.g., for signed b-bit integers, $Q_{\min} = -2^{b-1}$ and $Q_{\max} = 2^{b-1} - 1$).

2.2 LOW-PRECISION FLOATING-POINT FORMATS

Floating-point representation (Markstein, 2008) uses three fields: the sign bit (S), the exponent (E), and the mantissa (M). We denote a format as ExMy, where x and y are the numbers of exponent and mantissa bits. The sign determines the polarity, the exponent sets the dynamic range, and the mantissa sets the precision. A floating-point number decodes as:

$$\mathbb{C}_{\text{FP}} = \begin{cases} (-1)^S \times (1.M)_2 \times 2^{E-\text{bias}} & \text{if } E \neq 0 \text{ (Normal)}, \\ (-1)^S \times (0.M)_2 \times 2^{1-\text{bias}} & \text{if } E = 0,\ M \neq 0 \text{ (Subnormal)}. \end{cases} \tag{2}$$

Here, $\mathbb{C}_{\text{FP}}$ denotes the set of representable low-bit floating-point values. Floating-point quantization is:

$$\mathbf{X_q} = \text{Nearest}\left(\frac{\mathbf{X}}{s}, \mathbb{C}_{\text{FP}}\right) \cdot s, \tag{3}$$

where $\text{Nearest}(\cdot, \mathbb{C}_{\text{FP}})$ maps normalized values to the nearest element of $\mathbb{C}_{\text{FP}}$. Eq. (3) is a general quantization form that also recovers integer quantization by replacing $\mathbb{C}_{\text{FP}}$ with $\mathbb{C}_{\text{INT}}$.

Table 1: Low-bit formats name and their correspond represented range and scale factors.

Format	Block Size	Max Value	Min Value	Dynamic Range	Scale-1	Scale-2
MXFP8 (E4M3)	32	± 448	$\pm 2^{-9}$	1.75×2^{17}	UE8M0	-
MXINT8	32	127	1	127	UE8M0	-
MXFP6 (E2M3)	32	± 7.5	± 0.125	60	UE8M0	-
MXINT6	32	± 31	± 1	31	UE8M0	-
MXFP4 (E2M1)	32	± 6	± 0.5	12	UE8M0	-
MXINT4	32	± 7	± 1	7	UE8M0	-
NVFP4	16	± 6	± 0.5	12	E4M3	FP32
NVINT4	16	± 7	± 1	7	E4M3	FP32

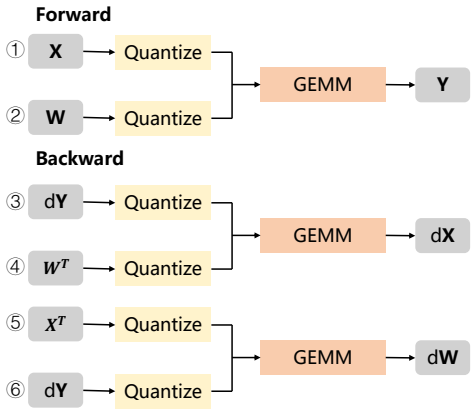


Figure 1: Compute flow of low-bit forward and backward propagation of linear layer.

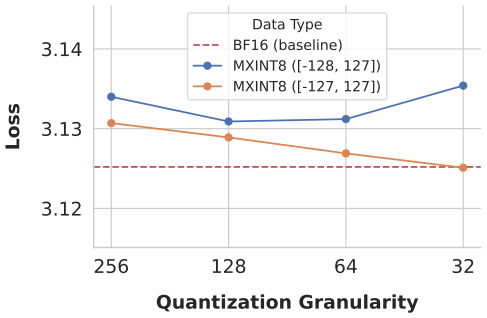


Figure 2: Impact of clipping range on INT8 final training loss on 145M model with 20B training tokens. Scale factor is kept on BF16 to emphasize the harm of asymmetric representation space during low-bit training.

2.3 Quantization Granularity

Quantization granularity specifies how scale factors apply across a tensor. Finer granularity usually improves accuracy but increases compute and memory overhead due to more scale factors. Common choices are: (i) Per-tensor: a single scale for the entire tensor. (ii) Per-channel: a scale per channel, broadcast along a chosen axis. (iii) Block-k: the tensor is partitioned into $1 \times k$ blocks along one dimension, and each block has its own scale. Block quantization is a key technique for improving accuracy at low precision. In this paper, we mainly focus on block quantization.

2.4 Block-Quantization Formats

To improve low-bit accuracy, OCP (Rouhani et al., 2023) proposes the Microscaling (MX) format, which uses a shared UE8M0[1] scale for each block of 32 elements. This fine-grained scaling reduces quantization error. Recently, NVIDIA Blackwell-series GPUs (NVIDIA Corporation, 2024b) provide native hardware support for MXFP8/MXFP6/MXFP4. Traditionally, FP8 has E4M3 and E5M2 variants, and FP6 has E2M3 and E3M2 variants. We consider E4M3 for MXFP8 and E2M3 for MXFP6 because mantissa bits are more critical to the performance of fine-grained quantization, consistent with prior work (Liu et al., 2024a; Mishra et al., 2025; Rouhani et al., 2023). Furthermore, NVIDIA proposes NVFP4, which enhances MXFP4 by reducing the block size from 32 to 16 and replacing the UE8M0 scale with an E4M3 scale. NVFP4 also introduces a second-level per-tensor scale to prevent overflow of the first-level E4M3 scale. Therefore, current hardware tends to support low-bit fine-grained floating-point formats. To enable fair comparison between low-bit floating-point and integer formats, we also introduce four corresponding integer variants: MXINT8, MXINT6, MXINT4, and NVINT4. Details of these low-bit formats are listed in Table 1.

[1]UE8M0 is an 8-bit unsigned floating-point format with eight exponent bits and zero mantissa bits.

3 QUANTIZATION RECIPE

This section illustrates the computation flow for low-bit inference and training in Sec. 3.1, and details the scale-factor computation used in quantization in Sec. 3.2.

3.1 QUANTIZATION COMPUTE FLOW

Figure 1 shows an example of using low-bit GEMM in a linear layer during forward and backward propagation. Given high-precision (e.g., BFloat16) activations $\mathbf{X}$ and weights $\mathbf{W}$, the forward pass of the quantized linear layer[2] is:

$$\mathbf{Y} = \underbrace{\text{Quantize}(\mathbf{X})}_{①}\ \underbrace{\text{Quantize}(\mathbf{W})}_{②}. \tag{4}$$

The backward pass to compute $d\mathbf{X}$ and $d\mathbf{W}$ is:

$$d\mathbf{X} = \underbrace{\text{Quantize}(\mathbf{dY})}_{③}\ \underbrace{\text{Quantize}(\mathbf{W}^T)}_{④}, \tag{5}$$

$$d\mathbf{W} = \underbrace{\text{Quantize}(\mathbf{X}^T)}_{⑤}\ \underbrace{\text{Quantize}(\mathbf{dY}^T)}_{⑥}. \tag{6}$$

Quantize$(\cdot)$ maps high-precision tensors to low-bit representations. Thus, there are six quantization operations in one linear layer: ① $\mathbf{X}$ and ② $\mathbf{W}$ in Eq. (4); ③ $\mathbf{dY}$ and ④ $\mathbf{W}^T$ in Eq. (5); ⑤ $\mathbf{X}^T$ and ⑥ $\mathbf{dY}^T$ in Eq. (6). Block-wise quantization requires tensors to be quantized along the GEMM reduction dimension to gain hardware benefits. Therefore, ① and ⑤, ② and ④, and ③ and ⑥ are quantized along different axes (Liu et al., 2024a; Darvish Rouhani et al., 2023). We separately analyze the quantization error of these six operations in Sec. 5.3.

3.2 QUANTIZATION OPERATION

UE8M0 scale factor. The scale factor s in Eq. (1) and Eq. (3) is computed with the AbsMax quantizer:

$$s = \frac{\text{AbsMax}(\mathbf{X})}{Q_{max}}, \tag{7}$$

where $\text{AbsMax}(\mathbf{X})$ is the maximum absolute value within the group of values that share a single scale factor, and Q_{max} is the maximum value of the quantized type (see Table 1). Eq. (7) maps the largest magnitude in high precision to the maximum representable low-precision value without clipping. OCP (Rouhani et al., 2023) further converts the high-precision scale factor to the UE8M0 format for MX formats:

$$s' = \text{clip}(\lfloor \log_2(\text{AbsMax}(\mathbf{X})) \rfloor - \lfloor \log_2(Q_{max}) \rfloor, -127, 127), \tag{8}$$

where $\lfloor \cdot \rfloor$ denotes rounding down. Eq. (8) rounds the high-precision scale down to the nearest UE8M0 value, which introduces extra clipping error. Following existing works (Tseng et al., 2025; Chen et al., 2025b; Mishra et al., 2025), we round up the UE8M0 scale based on Eq. (7) to avoid this error:

$$s' = \text{clip}(\lceil \log_2(s) \rceil, -127, 127), \tag{9}$$

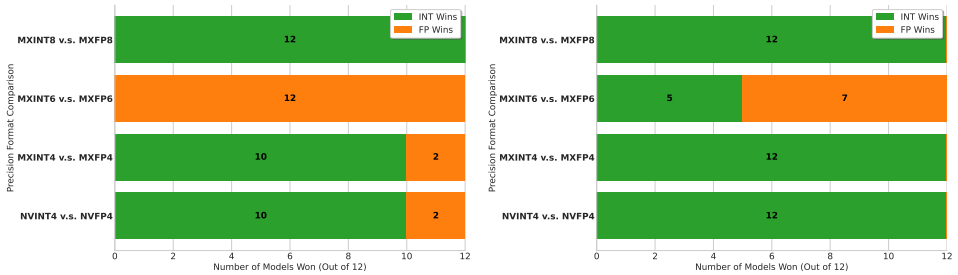
where $\lceil \cdot \rceil$ denotes rounding up.

Symmetric Clipping. Floating-point formats are naturally symmetric around zero. In contrast, signed integers in two's complement have one extra negative value: for a b-bit integer, $Q_{min} = -2^{b-1}$ and $Q_{max} = 2^{b-1} - 1$ (NVIDIA Corporation, 2024b). We find that this asymmetric range usually does not affect inference. However, as shown in Figure 2, it degrades INT8 training due to a persistent negative bias in gradients. Finer-grained quantization suffers more because more values fall into the unique negative endpoint Q_{min}. For INT8, the minimum value in a group can still map to -128 even when the scale is set to $\text{AbsMax}(\mathbf{X})/127$ due to BFloat16 arithmetic precision (see Sec. D.2 for details). Therefore, we use a symmetric integer range for all INT quantizers:

$$Q_{min} = -(2^{b-1} - 1), \quad Q_{max} = 2^{b-1} - 1,$$

as shown in Table 1.

[2] We omit the bias term.



(a) Direct-cast inference. (b) Direct-cast inference(w/ Hadamard rotation).

Figure 3: INT v.s. FP on 12 models through KL divergence on WikiText2. (a) shows the results with direct-cast inference, while (b) shows the results with direct-cast inference combined with random Hadamard rotation. Detailed numbers can be found in Sec. D.3.

4 FP *v.s.* INT

In this section, we compare low-bit integer and floating-point formats for both inference and training. For inference, we quantize the forward GEMMs (① and ② in Figure 1) of linear layers. For training, we quantize all GEMMs, including forward and backward (① to ⑥ in Figure 1), of linear layers.

4.1 Direct-Cast Inference

Precisions. For inference, we compare all data formats in Table 1: MXFP8, MXINT8, MXFP6, MXINT6, MXFP4, MXINT4, NVFP4, and NVINT4. We perform quantized inference on a trained BFloat16 model, and we quantize all forward GEMMs.

Models. We evaluate LLMs across a wide range of sizes, including dense and Mixture-of-Experts (MoE) models, from 0.6B to 235B parameters. The evaluated models include Qwen3-0.6B/1.7B/4B/8B/14B/32B/30B-A3B/235B-A22B (Yang et al., 2025), and Llama-3.1-8B/70B and Llama-3.2-1B/3B (Dubey et al., 2024). We provide the official open-source links in Sec. D for reproduction.

Metrics. Our goal is to compare integer and floating-point low-bit formats under the same setting, so ranking is more informative than absolute accuracy. Following Dutta et al. (2024), accuracy alone is not sufficient for compressed models because it can hide large changes in behavior. We therefore evaluate quantized models with distance metrics. Specifically, we compute the KL divergence on WikiText2 (Merity et al., 2016) between each quantized model and its BFloat16 counterpart. To reduce noise, we compute the divergence over the softmax distributions restricted to the top-25 logits of the BFloat16 model.

Results of direct-cast inference. As shown in Figure 3a, MXINT8 surpasses MXFP8 in all 12 models. MXINT4 and NVINT4 also outperform their floating-point counterparts, MXFP4 and NVFP4, in 10 of 12 models. In contrast, for 6-bit formats the trend reverses: MXFP6 outperforms MXINT6 in all 12 models.

Results of direct-cast inference with Hadamard rotation. Random Hadamard rotation (Ashkboos et al., 2024) is a popular technique to smooth distributions before quantization (Tseng et al., 2025; Chen et al., 2024a). We therefore rotate the inputs and weights, and quantize $\mathbf{XR}$ and $\mathbf{R}^T\mathbf{W}$, where $\mathbf{R}$ is a random Hadamard matrix of size $h \times h$. We set h equal to the block size (32 for MX formats and 16 for NV formats). As shown in Figure 3b, integer quantization benefits more from this outlier alleviation. Specifically, MXINT8, MXINT4, and NVINT4 outperform their FP counterparts in all 12 evaluated models, and the winning rate of MXINT6 improves from 0 to 5/12.

4.2 Training

Precisions. For training, we focus on nearly lossless low-bit training, which is more practical. Therefore, we study only the 8-bit setting and compare MXINT8 and MXFP8, since FP8 training is demonstrated to be nearly lossless in prior work (Mishra et al., 2025; Liu et al., 2024a).

Table 2: **Low-bit training comparisons.** HS, OB, and WG represents Hellaswag, OpenbookQA, and WinoGrande, respectively.

Model size	Training tokens	Precision	loss	Arc_E	Arc_C	HS	OB	PIQA	WG	Avg.
1B	100B	BF16	2.6727	37.80	69.40	60.20	38.40	74.43	61.09	56.89
1B	100B	MXFP8	2.6767	37.03	69.82	60.28	38.00	74.37	61.64	56.86
1B	100B	MXINT8	**2.6758**	37.95	69.45	60.02	38.80	74.54	61.38	**57.02**
3B	200B	BF16	2.4794	46.50	75.42	72.28	45.00	78.07	69.45	64.45
3B	200B	MXFP8	2.4821	46.70	74.12	72.08	44.60	77.56	69.25	64.05
3B	200B	MXINT8	**2.4812**	46.10	75.58	72.00	44.80	77.78	69.55	**64.30**

Models and datasets. We train 1B and 3B Llama3-style (Dubey et al., 2024) models on the OLMo2-Mix-1124 (OLMo et al., 2024) pretraining dataset, with 100B and 200B training tokens, respectively. Detailed model architectures and training hyperparameters are in Sec. D.

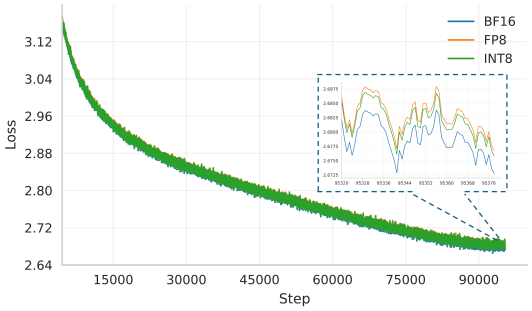


Figure 4: Loss curves comparison among BF16, MXFP8 and MXINT8 training on Llama-1B with 100B tokens. Results are smoothed by exponential moving average with a coefficient of 0.9.

Metrics. We measure training performance using two metrics: training loss and task accuracy. We smooth the training loss with an exponential moving average (coefficient 0.9). We compute all accuracies with `lm_eval` (Gao et al., 2024) through 5-shot evaluation. We report `acc` for WinoGrande (Sakaguchi et al., 2021) and `acc_norm` for HellaSwag (Zellers et al., 2019), Arc_Challenge, Arc_Easy (Clark et al., 2018), PIQA (Bisk et al., 2020), and Openbookqa (Mihaylov et al., 2018).

Results. Figure 4 shows the loss curves for BF16, MXFP8, and MXINT8 training. The curves for MXFP8 and MXINT8 almost overlap with BF16. In addition, MXINT8 consistently outperforms MXFP8 with a loss that is lower by approximately 0.001, as shown in the enlarged view in Figure 4. Table 2 shows that MXINT8 also achieves nearly the same average accuracy across six commonsense reasoning tasks compared to BF16 training. These results demonstrate that MXINT8 supports nearly lossless low-bit training, while existing works (Liu et al., 2024a; Mishra et al., 2025) mainly focus on FP8 training.

5 Deep Analysis: Why Fine-Grained INT Excels?

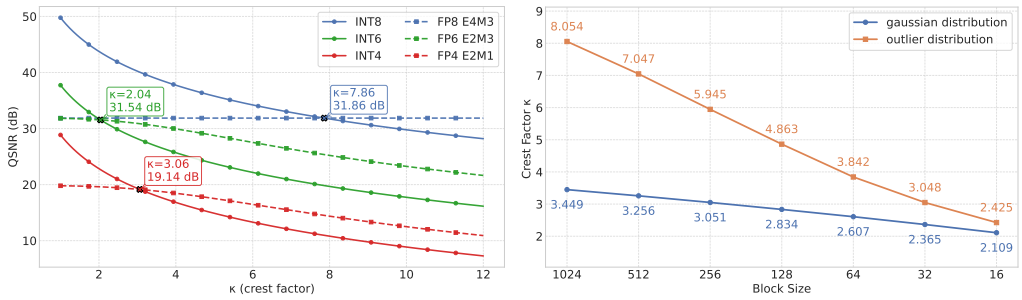
In this section, we provide an in-depth analysis of low-bit integer and floating-point formats. Specifically, Sec. 5.2 provides a theoretical comparison from the perspective of quantization error, and Sec. 5.3 validates this theory with empirical quantization data.

5.1 Metrics

In our analysis, we use Quantization Signal-to-Noise Ratio (QSNR, dB) (Darvish Rouhani et al., 2023) to measure numerical fidelity under different quantization schemes. QSNR is the ratio of the power of the original signal $\mathbf{X}$ to the power of the quantization noise $\mathbf{X} - \mathbf{X}_q$, expressed in decibels:

$$\text{QSNR} = -10 \log_{10} \left(\frac{\|\mathbf{X} - \mathbf{X}_q\|^2}{\|\mathbf{X}\|^2} \right). \tag{10}$$

A higher QSNR indicates that the quantized vector better preserves the direction and magnitude of the original vector.



(a) QSNR comparisons across crest factor κ.

(b) Crest factor decrease with smaller block size.

Figure 5: **Theoretical analysis of quantization performance.** (a) QSNR comparison between various integer (INT) and floating-point (FP) formats across a range of crest factors (κ), derived from Eq. (11) and Eq. (12). The boxes represent the crest factor and QSNR of the intersection point of the INT and FP curves. (b) The effect of quantization block size on the average crest factor for a gaussian distribution and an outlier-prone distribution from Llama-3.1-8B activations.

5.2 Theoretical Analysis

Common assumptions. We consider block vectors $\mathbf{X} \in \mathbb{R}^k$ with i.i.d. entries $X_i \sim \mathcal{N}(0, \sigma^2)$. The block root-mean-equare (RMS) is σ, and the **block crest factor** is $\kappa := \frac{\max(|\mathbf{X}|)}{\sigma}$. We use blockwise AbsMax scaling with a power-of-two deployed scale $s' = \rho\, s$, where $s = \max(|\mathbf{X}|)$ and $\rho \in [1, 2)$ is the overhead from the power-of-two constraint. We choose $s' \geq s$ following Eq. (9) to avoid clipping at the upper bound. Throughout we adopt Bennett's high-resolution model (Bennett, 1948), i.e., within-cell errors are unbiased and approximately independent of the signal. The choice $s' \geq s$ applies to both INT and FP quantization.

Theorem 1 (INT QSNR). Under b-bit INT quantization with AbsMax and power-of-two scaling, the QSNR (in dB) is

$$\boxed{\mathrm{QSNR} \;\approx\; 4.78 \;+\; 6.02\, b \;-\; 20 \log_{10}(\rho) \;-\; 20 \log_{10}(\kappa)} \tag{11}$$

A detailed proof of Theorem 1 is given in Sec. B.2, where b is the bit width, $\rho \in [1, 2)$ is the scale overhead, and $\kappa = \max(|\mathbf{X}|)/\mathrm{RMS}(\mathbf{X})$ is the crest factor.

Interpretation of theorem 1. (i) In the high-resolution, no-clipping regime, each additional bit yields ≈ 6.02 dB. (ii) The power-of-two constraint costs up to $20 \log_{10}(\rho) \leq 6.02$ dB. (iii) A larger crest factor κ degrades QSNR; smaller quantization block sizes typically reduce κ and improve QSNR.

Theorem 2 (FP QSNR). Under FP quantization with AbsMax and power-of-two scaling, the QSNR (in dB) is

$$\boxed{\mathrm{QSNR} \;=\; -10 \log_{10}\!\big(\alpha_M\, w_{\mathrm{norm}} \;+\; \beta\, (\rho\, \kappa)^2\, p_{\mathrm{sub}} \;+\; w_{\mathrm{zero}}\big)} \tag{12}$$

A detailed proof of Theorem 2 is given in Sec. B.3, with $\alpha_M = \frac{1}{24 \cdot 2^{2M}}$ (mantissa-resolution term) and $\beta = \frac{2^{2(1-B-M)}}{12\, Q_{\max}^2}$. Here M is the mantissa bit width, B is the exponent bias, and $Q_{\max}$ is the largest finite normal magnitude of the target FP format (e.g., $Q_{\max} = 448$ for E4M3). The terms $w_{\mathrm{norm}}, p_{\mathrm{sub}}$, and w_{zero} quantify how much of the numbers falls into the normal, subnormal, and zero regions (after scaling): w_{norm} is the fraction of signal energy represented by normal FP numbers and incurs mantissa quantization error α_M; p_{sub} is the probability mass encoded as subnormals and incurs a fixed absolute step error whose magnitude grows with $(\rho\kappa)$ via $\beta(\rho\kappa)^2$; and w_{zero} is the fraction of energy that underflows to zero.

Interpretation of theorem 2. (i) The mantissa bit width sets the upper bound on FP QSNR. With ample dynamic range ($w_{\mathrm{norm}} \approx 1$ and $p_{\mathrm{sub}} \approx w_{\mathrm{zero}} \approx 0$), QSNR $\approx 13.80 + 6.02\, M$ dB, independent of block granularity and the distribution of $\mathbf{X}$. (ii) A larger crest factor κ increases the share of

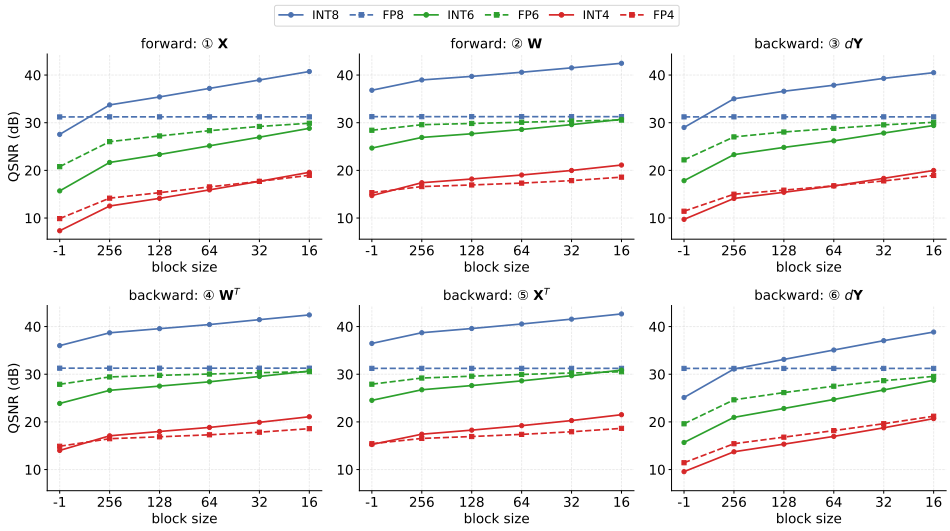


Figure 6: QSNR versus block size for quantization across 6 forward/backward items. -1 denotes per-channel quantization. 8-bit and 6-bit use E8M0 scales for MX-format, and 4-bit uses E4M3 scales for NV-format. Detailed numbers can be find in Table. 14.

subnormals and zeros, which degrades QSNR. Finer-grained blocks reduce κ, lower p_{sub} and w_{zero}, and improve QSNR.

Theoretical comparisons. With Eq. (11) in Theorem 1 and Eq. (12) in Theorem 2, we estimate the QSNR of low-bit integer and floating-point formats for a given bit width and target distribution (via κ). Specifically, we set $\rho = 1.44$ to imitate MX-formats and $\rho = 1.0$ to imitate NV-formats. As shown in Figure 5a, we observe:

- **INT8 *vs.* FP8**: FP8 QSNR varies smoothly due to its ample dynamic range. INT8 outperforms FP8 when $\kappa < 7.86$.
- **INT6 *vs.* FP6**: FP6 has the same QSNR as FP8 at small κ, because both FP6 and FP8 have three mantissa bits. However, FP6 QSNR decreases rapidly as κ increases due to limited dynamic range. INT6 outperforms FP6 only when $\kappa < 2.04$.
- **INT4 *vs.* FP4**: INT4 outperforms FP4 when $\kappa < 3.06$.

Furthermore, Figure 5b shows that the crest factor κ decreases as block size decreases. For a Gaussian distribution, κ decreases from 3.449 at block size 1024 to 2.365 at block size 32. For an outlier-heavy distribution, κ decreases from 8.054 at block size 1024 to 3.048 at block size 32. Since 3.048 lies below the 8-bit/4-bit intersection in Figure 5a, this explains why MXINT8, MXINT4, and NVINT4 outperform their floating-point counterparts in most cases, as shown in Figure 3a. However, MXINT6 lags behind MXFP6 because it only outperforms when $\kappa < 2.04$, whereas even a Gaussian distribution has $\kappa = 2.365 > 2.04$ at block size 32. In addition, the benefit of integer quantization increases as κ decreases, so outlier-alleviation techniques (Ashkboos et al., 2024; Liu et al., 2024b; Shao et al., 2023; Chen et al., 2024a) can further improve integer performance relative to floating-point quantization as demonstrated in Figure 3b.

5.3 Statistical Analysis

Setup. To measure the QSNR in real data, we feed 8 WikiText2 (Merity et al., 2016) sequences of length 4096 into Llama-3.1-8B, run both forward and backward propagation in BFloat16 precision, and capture the six intermediate tensors (weights, activations, and gradients) indicated by ①–⑥ in Figure 1. Llama-3.1-8B contains 224 linear layers across all transformer blocks. We collect these tensors for all 224 linear layers and use them to compute the QSNR under different quantization formats. Because tensors of the same type have similar distributions, we report the QSNR averaged within each of the six types separately. Specifically, we evaluate INT and FP quantization at 8, 6,

and 4 bits. We use E8M0 scales for MX-format at 8 and 6 bits, and E4M3 scales for NV-format at 4 bits, since NV-format significantly outperforms MX-format in the 4-bit setting.

Results. Figure 6 reports measured QSNR (dB) across six tensor types and block sizes for INT/FP at 8/6/4 bits. The empirical trends closely follow the theoretical comparisons in Sec. 5.2 (Theorems 1–2) and the crest-factor analysis:

- **FP is mantissa-limited and block-size invariant.** FP8/FP6 curves are nearly flat across block sizes (e.g., FP8 ≈ 31.2 dB; FP6 ≈ 26–30.6 dB), consistent with Theorem 2(i).
- **INT benefits from finer blocks via reduced crest factor.** INT8/INT6/INT4 QSNR increases monotonically as blocks shrink (e.g., INT8 rises to $\sim$ 37–41 at block size as 32), matching the κ-dependence in Theorem 1 and Fig. 5b.
- **Crossovers match predicted κ thresholds.**
  - **INT8 vs. FP8:** INT8 exceeds FP8 for most tensors at practical block sizes less than 256.
  - **INT6 vs. FP6:** FP6 generally dominates; INT6 only ties or slightly surpasses FP6 at the smallest blocks (GS$=16$) for weight-centric and columnwise activations—consistent with the tighter κ threshold for 6-bit.
  - **INT4 vs. FP4:** INT4 overtakes FP4 for most tensors once blocks size are less than 32 in most scenarios, while ⑥$d\mathbf{Y}$ remains slightly FP-favored, reflecting heavier tails.

Overall, real-data measurements corroborate the theory: FP QSNR is set by mantissa precision and largely insensitive to block granularity, whereas INT QSNR improves with smaller blocks as κ drops. This explains the strong performance of blockwise MXINT8 over MXFP8, the robustness of MXFP6 over MXINT6 except with outlier-alleviation techniques, and the superiority of MXINT4 and NVINT4 relative to MXFP4 and NVFP4 for most cases.

Table 3: Normalized energy and area costs of low-bit formats at matched throughput. Single-format results use MXFP8 as the baseline, and mixed-format results use MXFP8+MXFP4 as the baseline.

	Single Format				Mixed Format	
	MXFP8	MXINT8	MXFP4	MXINT4	MXFP8+MXFP4	MXINT8+MXINT4
Energy	1x	**0.67x**	0.31x	**0.23x**	1x	**0.74x**
Area	1x	**0.83x**	0.29x	**0.27x**	1x	**0.70x**

6 Hardware Cost Analysis

Based on the hardware model in Sec. C, we evaluate the energy and area cost of a Matrix-Multiply Unit (MMU) that supports the MX format. Table 3 shows that MXINT8 and MXINT4 reduce energy by 33% and 26%, respectively, compared with MXFP8 and MXFP4. We also evaluate mixed-format configurations. Following the NVIDIA Blackwell GPUs (NVIDIA Corporation, 2024b), we study a chip that supports both 8-bit and 4-bit data types and set the throughput ratio of 8-bit to 4-bit to 1:2 to match the communication bandwidth. As shown in Table 3, the "MXINT8+MXINT4" configuration further reduces area by about 30% relative to "MXFP8+MXFP4", mainly because circuit reuse is simpler in the INT pipeline (Table 5). Overall, this analysis shows that, at matched throughput, low-bit integer formats are more hardware-efficient than low-bit floating-point formats.

7 Conclusion

Our comprehensive study reveals a critical crossover point where fine-grained integer (INT) quantization consistently outperforms floating-point (FP) formats for modern LLMs. This finding challenges the current hardware trajectory, as we show INT formats provide a dual advantage of superior accuracy and greater hardware efficiency. We therefore call for a strategic shift in both academia and industry toward algorithm-hardware co-design centered on fine-grained INT to build more powerful and efficient AI accelerators.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

W. R. Bennett. Spectra of quantized signals. *Bell System Technical Journal*, 27(3):446–472, July 1948. doi: 10.1002/j.1538-7305.1948.tb01364.x.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

Roberto L Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native fp4 training can be optimal for large language models. *arXiv preprint arXiv:2505.14669*, 2025.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Eliminating outliers by prefixed tokens for large language models quantization. *arXiv preprint arXiv:2410.05265*, 2024a.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*, 2024b.

Mengzhao Chen, Chaoyi Zhang, Jing Liu, Yutao Zeng, Zeyue Xue, Zhiheng Liu, Yunshui Li, Jin Ma, Jie Huang, Xun Zhou, et al. Scaling law for quantization-aware training. *arXiv preprint arXiv:2505.14302*, 2025a.

Yuxiang Chen, Haocheng Xi, Jun Zhu, and Jianfei Chen. Oscillation-reduced mxfp4 training for vision transformers. *ArXiv*, abs/2502.20853, 2025b.

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

Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, et al. With shared microexponents, a little shifting goes a long way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–13, 2023.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Elias Frantar, Utku Evci, Wonpyo Park, Neil Houlsby, and Dan Alistarh. Compression scaling laws: Unifying sparsity and quantization. *arXiv preprint arXiv:2502.16440*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Tanishq Kumar, Zachary Ankner, Benjamin F Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. *arXiv preprint arXiv:2411.04330*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

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. *arXiv preprint arXiv:2405.16406*, 2024b.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Scaling laws in extremely low-bit llm quantization. *arXiv preprint arXiv:2502.02631*, 2025.

Peter Markstein. The new ieee-754 standard for floating point arithmetic. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2008.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Asit Mishra, Dusan Stosic, and Simon Layton. Recipes for pre-training llms with mxfp8. *arXiv preprint arXiv:2506.08027*, 2025.

Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman Jouppi, and David Patterson. The design process for google's training chips: Tpuv2 and tpuv3. *IEEE Micro*, 41(2):56–63, 2021. doi: 10.1109/MM.2021.3058217.

NVIDIA Corporation. Nvidia a100 tensor core gpu architecture. Whitepaper, NVIDIA Corporation, 2020. URL https://www.nvidia.com/en-us/data-center/ampere-architecture/.

NVIDIA Corporation. Nvidia h100 tensor core gpu architecture. Whitepaper, NVIDIA Corporation, 2022. URL https://www.nvidia.com/en-us/data-center/hopper-architecture/.

NVIDIA Corporation. Nvidia blackwell gpu architecture. Whitepaper, NVIDIA Corporation, 2024a. URL https://www.nvidia.com/en-us/data-center/blackwell-architecture/.

NVIDIA Corporation. Working with quantized types – nvidia tensorrt documentation. https://docs.nvidia.com/deeplearning/tensorrt/latest/inference-library/work-quantized-types.html, 2024b. Accessed: 2025-09-03.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

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

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

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. *arXiv preprint arXiv:2502.20586*, 2025.

Sami Ul Haq, Aiman H. El-Maleh, and Ali Alsuwaiyan. Multiple-input floating-point adders: A comprehensive review. *IEEE Access*, 13:91012–91024, 2025. doi: 10.1109/ACCESS.2025.3572430.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Yijia Zhang, Lingran Zhao, Shijie Cao, Sicheng Zhang, Wenqiang Wang, Ting Cao, Fan Yang, Mao Yang, Shanghang Zhang, and Ningyi Xu. Integer or floating point? new outlooks for low-bit quantization on large language models. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2024.

OUTLINES

- Sec. A introduces related works.
- Sec. B details the proofs of Theorems 1 and 2 on INT and FP QSNR estimation.
- Sec. C presents the hardware cost estimation model.
- Sec. D provides additional details on the models used and ablation studies, and reports the numerical results corresponding to the figures in the main paper.

USAGE OF LARGE LANGUAGE MODELS

We use LLMs to polish the paper, correct the grammar, and for some of the figures in the article, the initial drawing codes are generated by LLMs.

A RELATED WORK

Quantization Algorithms. Quantization methods include post-training quantization (PTQ) (Lin et al., 2023; Frantar et al., 2022; Shao et al., 2023; Xiao et al., 2023) and quantization-aware training (QAT) (Chen et al., 2024b; Liu et al., 2025), which speed up inference. Low-bit training (Mishra et al., 2025; Tseng et al., 2025; Chen et al., 2025b) speeds up both training and inference. Several works also study scaling laws (Hoffmann et al., 2022) for low-bit quantization (Castro et al., 2025; Chen et al., 2025a; Frantar et al., 2025; Kumar et al., 2024). However, most prior work focuses on a single low-bit format—either integer or floating-point—and does not provide direct comparisons between these formats. Zhang et al. (2024) study mixed-format quantization in the PTQ setting, assigning integer or floating-point formats to different model parts.

Hardware. Previous accelerators (NVIDIA Corporation, 2020; 2022) do not natively support fine-grained quantization, so algorithms (Xiao et al., 2023; Chen et al., 2024a) face challenges with per-channel quantization in the presence of outliers (Sun et al., 2024). Recently, OCP Rouhani et al. (2023) proposes Microscaling (MX) data formats, which combine a per-block scaling factor with a block size of 32 to improve low-bit quantization performance. NVIDIA Blackwell (NVIDIA Corporation, 2024a) supports MXFP8, MXFP4, and NVFP4 at the hardware level.

B PROOFS OF THEOREMS

B.1 COMMON ASSUMPTIONS AND NOTATION

We consider block vectors $\mathbf{X} \in \mathbb{R}^k$ with i.i.d. entries $X_i \sim \mathcal{N}(0, \sigma^2)$. Let the block RMS be $\sigma := \mathrm{RMS}(\mathbf{X})$ and the block crest factor

$$\kappa \;:=\; \frac{\max(|\mathbf{X}|)}{\sigma}.$$

We adopt Bennett's high-resolution model (Bennett, 1948): within-cell errors are (approximately) unbiased, uniform, and independent of the signal.

We use blockwise AbsMax scaling with a power-of-two deployed scale

$$s' \;=\; 2^{\lceil \log_2 s \rceil} \;=\; \rho\, s, \qquad \rho \in [1, 2),$$

and choose $s' \geq s$ to avoid upper clipping. The ideal scale s is chosen so that the largest codebook magnitude in the target format matches the block maximum after scaling:

$$s \;=\; \frac{\max(|\mathbf{X}|)}{Q_{\mathrm{ref}}},$$

with Q_{ref} determined by the target format:

- INT(b) (symmetric): $Q_{\mathrm{ref}} = Q := 2^{b-1} - 1$ (largest integer code).
- FP(E, M, B) (with subnormals): $Q_{\mathrm{ref}} = Q_{\max}$ (largest finite normal magnitude of the FP format; e.g., $Q_{\max} = 448$ for E4M3).

This convention is consistent with the main text: we use the same symbols $(\sigma, \kappa, \rho, s, s')$, and $s' \geq s$ guarantees no overflow in both INT and FP quantization. Unless otherwise stated, expectations are taken over both the data and the quantization randomness, and large-k averages are used so that $\|\mathbf{X}\|^2 \approx k\sigma^2$.

B.2 THEOREM 1 (INT QUANTIZATION)

Setup and scaling. Consider a symmetric, mid-tread, uniform quantizer with bit-width b and integer range $[-Q, Q]$ where

$$Q \;=\; 2^{b-1} - 1 \quad \text{(e.g., } Q \in \{127, 31, 7\} \text{ for } b \in \{8, 6, 4\}\text{)}.$$

AbsMax scaling uses the ideal scale

$$s \;=\; \frac{\max(|\mathbf{X}|)}{Q} \;=\; \frac{\kappa\,\sigma}{Q},$$

and the deployed scale is constrained to a power of two,

$$s' \;=\; 2^{\lceil \log_2 s \rceil} \;=\; \rho\, s, \qquad \rho \in [1, 2).$$

Quantize–dequantize (round-to-nearest) is

$$\mathbf{X}_q \;=\; \mathrm{clamp}\big(\,\mathrm{round}(\tfrac{\mathbf{X}}{s'}),\, -Q,\, Q\big) \cdot s'.$$

Because $s' \geq s$, we have $|\mathbf{X}|/s' \leq |\mathbf{X}|/s \leq Q$, hence no clipping occurs and the clamp is inactive. The effective (uniform) step is $\Delta := s'$.

Error model. Define the elementwise error $e := \mathbf{X} - \mathbf{X}_q$. For a non-saturating symmetric mid-tread quantizer with round-to-nearest, $e \in [-\Delta/2,\, \Delta/2]$. Under the high-resolution (Bennett) approximation, the error is approximately uniform and independent of $\mathbf{X}$:

$$\mathbb{E}[e] = 0, \qquad \mathbb{E}[e^2] = \frac{\Delta^2}{12}.$$

QSNR. Define the blockwise QSNR as

$$\mathrm{QSNR} \;=\; -10 \log_{10}\!\left(\frac{\|\mathbf{X} - \mathbf{X}_q\|^2}{\|\mathbf{X}\|^2}\right).$$

For large k, $\mathbb{E}[\|\mathbf{X}\|^2] \approx k\sigma^2$ and $\mathbb{E}[\|\mathbf{X} - \mathbf{X}_q\|^2] \approx k\,\mathbb{E}[e^2] = k\Delta^2/12$, so

$$\mathrm{QSNR} \;\approx\; -10 \log_{10}\!\left(\frac{\Delta^2}{12\,\sigma^2}\right).$$

Expressing Δ via crest factor and scale overhead. With $\kappa := \max(|\mathbf{X}|)/\sigma$ and $s = \kappa\sigma/Q$, we have $s' = \rho s$ and hence

$$\Delta \;=\; s' \;=\; \frac{\rho\,\kappa\,\sigma}{Q}.$$

Substituting into the QSNR expression gives

$$\frac{\Delta^2}{12\,\sigma^2} \;=\; \frac{(\rho\,\kappa)^2}{12\,Q^2},$$

and therefore

$$\boxed{\mathrm{QSNR} \;\approx\; 10 \log_{10}(12) \;+\; 20 \log_{10}(Q) \;-\; 20 \log_{10}(\rho) \;-\; 20 \log_{10}(\kappa)} \tag{13}$$

Using $Q = 2^{b-1} - 1$, this is well-approximated by

$$\boxed{\mathrm{QSNR} \;\approx\; 4.78 \;+\; 6.02\,b \;-\; 20 \log_{10}(\rho) \;-\; 20 \log_{10}(\kappa)} \tag{14}$$

making explicit: (i) ≈ 6.02 dB per additional bit, (ii) up to 6.02 dB loss from the power-of-two overhead ($\rho \in [1, 2)$), and (iii) a penalty scaling with the crest factor κ (which typically increases with larger block size).

B.3 THEOREM 2 (FP QUANTIZATION)

Setup and codebook. Let $\mathbf{X} \in \mathbb{R}^k$ be zero-mean with RMS σ and crest factor $\kappa := \max(|\mathbf{X}|)/\sigma$. Consider a target floating-point format FP(E, M, B) with sign, E exponent bits (bias B), and M mantissa bits, with subnormals enabled. In the normalized domain,

$$\text{normal: } x = \pm(1.f)_2\, 2^{e-B}, \quad \text{subnormal: } x = \pm(0.f)_2\, 2^{1-B}, \quad f \in \{1, \dots, 2^M - 1\}.$$

Let $Q_{\max}$ denote the largest finite normal magnitude (e.g., $Q_{\max} = 448$ for E4M3), $N_{\min} = 2^{1-B}$ (smallest normal), and $S_{\min} = 2^{1-B-M}$ (smallest nonzero subnormal).

Scaling and quantization rule. Per-block AbsMax scaling uses

$$s \;=\; \frac{\max(|\mathbf{X}|)}{Q_{\max}} \;=\; \frac{\kappa\,\sigma}{Q_{\max}}, \qquad s' \;=\; 2^{\lceil \log_2 s \rceil} \;=\; \rho\, s, \;\; \rho \in [1, 2).$$

Quantize–dequantize with round-to-nearest:

$$\mathbf{X}_q \;=\; s' \cdot \mathrm{Nearest}\Big(\tfrac{\mathbf{X}}{s'}, \mathbb{C}_{\mathrm{FP}}\Big),$$

where $\mathbb{C}_{\mathrm{FP}}$ is the FP codebook in the normalized domain. Since $s' \geq s$, no overflow occurs: $|\mathbf{X}|/s' \leq |\mathbf{X}|/s \leq Q_{\max}$. Underflow to subnormals or to zero is possible.

Error decomposition. Let $e := \mathbf{X} - \mathbf{X}_q$. We analyze the relative MSE

$$R \;:=\; \frac{\mathbb{E}[e^2]}{\mathbb{E}[X^2]} \;=\; \frac{\mathbb{E}[e^2]}{\sigma^2}, \qquad \mathrm{QSNR} \;:=\; -10 \log_{10} R.$$

Under a high-resolution (Bennett) model, within-cell error is unbiased and uniform on $[-\Delta/2, \Delta/2]$, and the logarithmic phase

$$r \;:=\; 2^{\{\log_2(|X|/s')\}} \in [1, 2)$$

(the fractional part $\{\cdot\}$ of $\log_2(|X|/s')$) is approximately uniform on $[1, 2)$.

Define signal-domain thresholds and the subnormal spacing

$$T_N := s' N_{\min}, \qquad T_0 := s'\,\frac{S_{\min}}{2} \quad \text{(round-to-nearest)}, \qquad \Delta_{\mathrm{sub}} := s'\, S_{\min} = s'\, 2^{1-B-M}.$$

We split the amplitude axis into three regions:

- **Normal region ($|X| \geq T_N$).** Let $e(X) := \lfloor \log_2(|X|/s') \rfloor$ be the exponent bin of X/s'. The local ULP is
$$\Delta(X) \;=\; s'\, 2^{\,e(X)-M}.$$
Writing $2^{e(X)} = |X|/(s'r)$ with $r \in [1, 2)$ gives
$$\Delta(X) \;=\; \frac{|X|}{r}\, 2^{-M}.$$
Uniform-error modeling yields $\mathbb{E}[e^2 \mid X, |X| \geq T_N] = \Delta(X)^2/12 = |X|^2\, 2^{-2M}/(12\, r^2)$. Averaging over $r \sim \mathrm{Unif}[1, 2]$ gives $\mathbb{E}[1/r^2] = \int_1^2 r^{-2}\, dr = 1/2$, hence
$$\mathbb{E}[e^2 \cdot \mathbf{1}\{|X| \geq T_N\}] \;\approx\; \alpha_M\, \mathbb{E}[X^2 \cdot \mathbf{1}\{|X| \geq T_N\}], \quad \alpha_M := \frac{1}{24 \cdot 2^{2M}}.$$
- **Subnormal but nonzero region ($T_0 \leq |X| < T_N$).** Here the absolute spacing is constant, Δ_{sub}, so
$$\mathbb{E}[e^2 \mid T_0 \leq |X| < T_N] \;\approx\; \frac{\Delta_{\mathrm{sub}}^2}{12} \;=\; \frac{s'^2\, 2^{2(1-B-M)}}{12}.$$
Let $p_{\mathrm{sub}} := \mathbb{P}(T_0 \leq |X| < T_N)$. Then
$$\mathbb{E}[e^2 \cdot \mathbf{1}\{T_0 \leq |X| < T_N\}] \;\approx\; \frac{s'^2\, 2^{2(1-B-M)}}{12}\, p_{\mathrm{sub}}.$$

- **Zero region** ($|X| < T_0$)**.** Quantization rounds to zero; thus $e = X$ and

$$\mathbb{E}[e^2 \cdot \mathbf{1}\{|X| < T_0\}] \;=\; \mathbb{E}[X^2 \cdot \mathbf{1}\{|X| < T_0\}].$$

Summing the three contributions and normalizing by σ^2 yields

$$\frac{\mathbb{E}[e^2]}{\sigma^2} \;\approx\; \alpha_M \, w_{\text{norm}} \;+\; \beta \, (\rho\,\kappa)^2 \, p_{\text{sub}} \;+\; w_{\text{zero}},$$

where we define the dimensionless weights

$$w_{\text{norm}} := \frac{\mathbb{E}[X^2 \cdot \mathbf{1}\{|X| \geq T_N\}]}{\sigma^2}, \qquad w_{\text{zero}} := \frac{\mathbb{E}[X^2 \cdot \mathbf{1}\{|X| < T_0\}]}{\sigma^2},$$

and used $s'^2/\sigma^2 = (\rho\kappa)^2/Q_{\max}^2$ with

$$\beta := \frac{2^{2(1-B-M)}}{12\,Q_{\max}^2}.$$

Therefore

$$\boxed{\text{QSNR} \;\approx\; -10\log_{10}\!\big(\alpha_M \, w_{\text{norm}} \;+\; \beta \, (\rho\,\kappa)^2 \, p_{\text{sub}} \;+\; w_{\text{zero}}\big)} \tag{15}$$

Thresholds and interpretation. The thresholds that determine w_{norm}, w_{zero}, and p_{sub} are

$$T_N \;=\; s' N_{\min} \;=\; \sigma\,(\rho\,\kappa)\,\frac{N_{\min}}{Q_{\max}}, \qquad T_0 \;=\; s'\,\frac{S_{\min}}{2} \;=\; \sigma\,(\rho\,\kappa)\,\frac{S_{\min}}{2Q_{\max}}.$$

Thus block size affects QSNR through κ (and ρ) only via underflow-related terms. In the ample dynamic-range regime (T_0, T_N tiny so that $w_{\text{norm}} \approx 1$ and $p_{\text{sub}} \approx w_{\text{zero}} \approx 0$), the law simplifies to

$$\text{QSNR} \;\approx\; 10\log_{10}(24) \;+\; 20\,M\log_{10}(2) \;=\; 13.80\ \text{dB} \;+\; 6.02\,M\ \text{dB}, \tag{16}$$

independent of block granularity and the distribution of $\mathbf{X}$.

Table 4: Gate-complexity model for the MAC Unit with k lanes. Here x and y denote exponent and mantissa widths; for INT, x=0. The aligner width n is given by equation 17. "Main Cells" list dominant standard cells used in aggregation.

Sub-block	INT Mul	FP Mul	INT Add	FP Add	Main Cells
Multiplier	$k(x{+}y{+}1)^2$	$k(y{+}1)^2$	–	–	AND, FA, HA
Adder (mantissa/int)	–	–	$2k(x{+}y{+}1)$	kn	FA, HA
Exponent adder	–	kx	–	–	FA, HA
Exponent subtractor	–	–	–	kx	XOR, FA, HA
Comparator	–	–	–	kx	XOR, AND, OR
Aligner (barrel)	–	–	–	$k\,n\log_2 n$	MUX
Normalizer (shared)	–	–	–	$n\log_2 n$	MUX, OR

C Hardware Cost Modeling

Scope and assumptions. We develop a compact gate-level model to estimate the chip area and energy of a GEMM engine under low-precision formats. Specifically, a low-bit GEMM engine uses four components: a quantizer, a multiply-and-accumulate (MAC) unit, a dequantizer, and an FP32 accumulator. The proposed model accounts only for the MAC unit and a shared FP32 accumulator; the quantizer and dequantizer are excluded from all cost accounting. In MX formats, the VPU implements quantization by shift-and-round, and the accumulation pipeline can fuse dequantization as two 8-bit integer additions. We omit these blocks to isolate the cost driven by multiplication and accumulation. Unless otherwise stated, we take cell factors from a TSMC FinFET standard-cell library. We model only combinational logic; we ignore sequential elements, placement and routing, and interconnect to enable technology-aware, relative comparisons.

Throughput Ratio	MXINT8 : MXINT4 = 1 : 2
No reuse	1 * int8_MAC_unit + 2 * int4_MAC_unit
INT reuse scheme 1	1 * int8_MAC_unit + 1 * int4_MAC_unit
INT reuse scheme 2	2 * int8_(u)int4_MAC_unit
Throughput Ratio	MXFP8 : MXFP4 = 1 : 2
No reuse	1 * e4m3_MAC_unit + 2 * e2m1_MAC_unit
FP reuse scheme	1 * e4m3_MAC_unit + 1 * e2m1_MAC_unit

Table 5: Comparison of MAC unit configurations for different reuse schemes. Notes: (1) No reuse: Highest energy efficiency for INT8 and INT4, but greatest area wastage; (2) INT reuse scheme 1: Use int8 lane as an int4 path directly (set the 8-b input to XXXX_0000), a little more energy cost for INT4 but lower area cost; (3) INT reuse scheme 2: Use two int8×(u)int4 lanes to reconfigure int8 lane or int4 lane, a little more energy cost for both INT4 and INT8, but lowest area cost; (4) No reuse: Highest energy efficiency for FP8 and FP4, but greatest area wastage; (5) FP reuse scheme: Use fp8 lane as an fp4 path directly (set the 8-b input to S_00XX_X00), a little more energy cost for FP4 but lower area cost. We adopt INT reuse scheme 2 and FP reuse scheme to evaluate the area cost shown in Table 3.

Design choice: FP32 accumulation and MMU integration. A high-throughput Matrix-Multiply Unit (MMU), as in TPU-like designs (Norrie et al., 2021), integrates the multiply-and-accumulate datapath and downstream accumulation to improve performance and energy efficiency. To prevent error growth and preserve scalability, we accumulate in FP32. Under the same nominal bit width, FP multipliers are typically more area- and energy-efficient than INT multipliers, whereas FP adders are more expensive than INT adders due to exponent comparison/subtraction, mantissa alignment, and normalization (Zhang et al., 2024). With a uniform-alignment design Ul Haq et al. (2025), the normalizer count reduces to one shared instance across the k MAC lanes, and we divide its cost by k.

Mantissa aligner width. The mantissa aligner couples accuracy and cost: its bit width n affects numerical fidelity and hardware complexity. We set

$$n \;=\; \min\big(2^x + 2y,\; \texttt{psum\_bit\_width}\big), \tag{17}$$

where x and y denote exponent and mantissa widths, respectively (for INT formats, $x = 0$). In all evaluations we use $k = 32$ and $\texttt{psum\_bit\_width} = 24$.

MAC unit structure and sub-blocks. We model the MAC unit as a k-lane array. Each lane comprises one multiplier. The adders from all lanes are fused together to form a multi-input adder tree structure, incorporating FP-specific alignment and normalization logic. Table 4 reports the dominant logic count (up to constant factors) for the main sub-blocks, where "Main Cells" indicate the standard-cell types used for area/energy aggregation. For FP multiplication, we multiply only mantissas and include an exponent adder. For FP addition, we model exponent comparator/subtractor, a barrel aligner, a wide mantissa adder, and one shared normalizer. For INT, we set $x = 0$ in the expressions.

Area and energy aggregation for MAC. Let $\mathcal{S}$={Multiplier, Adder(mantissa/int), Exponent adder, Exponent subtractor, Comparator, Aligner(barrel), Normalizer(shared)} be the set of sub-block types, and $\mathcal{G} = \{\text{FA}, \text{HA}, \text{XOR}, \text{AND}, \text{OR}, \text{MUX}\}$ be the set of cell types with technology-dependent area and energy factors A_g and E_g obtained from the standard-cell library. Let τ_g be the toggle rate of cell g, which represents the average switching activity of the cell. In this work, we simplify the toggle rate factor by assuming that all gate cells have the same toggle rate, $\tau_g = \tau$, to reduce computational complexity and focus on the primary design trade-offs. Denote by $c_{s,g}(x, y, k, n)$ the count of cell $g \in \mathcal{G}$ in sub-block s induced by the chosen format and by n from Eq.(17). The MAC area and energy are

$$\text{Area}_{\text{MAC}} \;=\; \sum_{s\in\mathcal{S}} \sum_{g\in\mathcal{G}} c_{s,g}(x, y, k, n)\, A_g, \qquad \text{Energy}_{\text{MAC}} \;=\; \sum_{s\in\mathcal{S}} \sum_{g\in\mathcal{G}} c_{s,g}(x, y, k, n)\, E_g \tau_g. \tag{18}$$

FP32 accumulator model. We model the FP32 accumulator by its combinational logic counts c_g^{ACC32}, yielding

$$\text{Area}_{\text{ACC32}} = \sum_{g \in \mathcal{G}} c_g^{\text{ACC32}} A_g, \qquad \text{Energy}_{\text{ACC32}} = \sum_{g \in \mathcal{G}} c_g^{\text{ACC32}} E_g. \tag{19}$$

Total cost and per-lane reporting. The total MMU cost is

$$\text{Area}_{\text{MMU}} = \text{Area}_{\text{MAC}} + \text{Area}_{\text{ACC32}}, \qquad \text{Energy}_{\text{MMU}} = \text{Energy}_{\text{MAC}} + \text{Energy}_{\text{ACC32}}, \tag{20}$$

and, when we report per-lane figures, we divide the cost of shared blocks by k.

Summary. The hardware model includes only the MAC unit and the FP32 accumulator; the quantizer and dequantizer are excluded from the overhead calculation. Given a low-precision format with exponent/mantissa widths (x, y) (with x=0 for INT), a MAC array size k, an aligner cap `psum_bit_width` (setting n via Eq (17), and technology cell factors $\{A_g, E_g\}_{g \in \mathcal{G}}$ (plus the FP32-accumulator gate counts), the model predicts the area and energy of the MAC and accumulation stages. It captures the relative cost trends across INT/FP/MX formats at the same nominal bit width, the sensitivity to the aligner width n (critical for FP addition), and the effect of sharing both the normalizer and the FP32 accumulator across k lanes.

D More Details for Reproduction

D.1 Used Models

Table 6: Huggingface IDs of evaluation models in direct-cast inference.

Model Name	Huggingface ID
Qwen3-0.6B	Qwen/Qwen3-0.6B-Base
Qwen3-1.7B	Qwen/Qwen3-1.7B-Base
Qwen3-4B	Qwen/Qwen3-4B-Base
Qwen3-8B	Qwen/Qwen3-8B-Base
Qwen3-14B	Qwen/Qwen3-14B-Base
Qwen3-32B	Qwen/Qwen3-32B
Qwen3-30B-A3B	Qwen/Qwen3-30B-A3B-Instruct-2507
Qwen3-235B-A22B	Qwen/Qwen3-235B-22B-Instruct-2507
Llama-3.2-1B	meta-llama/Llama-3.2-1B
Llama-3.2-3B	meta-llama/Llama-3.2-3B
Llama-3.1-8B	meta-llama/Meta-Llama-3.1-8B
Llama-3.1-70B	meta-llama/Meta-Llama-3.1-70B

Models for inference evaluation. We list the Huggingface IDs of evaluated open-sourced model for better reproduction in Tabel 6. Note that we firstly choose the base model without supervise fine-tuning if it is open-sourced, For a model of a certain size, our selection principle is that if the base model is open source, we will first choose the base model; otherwise, we will select the model that has undergone SFT.

Models for training evaluation. We select the Llama-3 (Dubey et al., 2024) style model for our experiments due to its wide adoption. The Llama-3 style model employs Group Query Attention (GQA)(Ainslie et al., 2023) for the self-attention module and SwiGLU(Shazeer, 2020) for the feed-forward module. Table 7 presents the detailed architectural settings abd training hyper-parameters of the models used.

D.2 Necessity of Symmetric Integer Representation

Table 8 offer the ablation studies on representation range of INT8 quantization. We find that the bias in representation range would consistently degenerate INT8 training loss. For BFloat16 scale

Table 7: Llama-3 style Model architecture and training hyper-parameters.

Model Size	145M	1B	3B
Layers	12	16	28
Hidden Size	1024	2048	3072
FFN Hidden Size	3072	8192	8192
Attention Heads	16	32	24
KV Heads	4	8	8
Batch Size (# Sequence)	256	512	512
Max LR	1.0e-3	6e-4	6e-4
Min LR	$0.1 \times$ Max LR		
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.95$)		
Weight Decay	0.1		
Clip Grad Norm	1.0		
LR Schedule	Cosine		
Warmup Steps	500		
Sequence Length	2048		

Table 8: Ablation studies about the clipping range on INT8 quantization across quantization granularities, as well as BFloat16 and UE8M0 scale factors. We repot the 8-bit training loss (lower is better) on a 145M model with 20B training tokens. The baseline of BF16 training without quantization

	BF16 scale		UE8M0 scale	
	[-128, 127]	**[-127, 127]**	[-128, 127]	**[-127, 127]**
per-channel	**3.2544**	3.2560	**3.3602**	**3.4307**
256	3.1340	**3.1307**	3.1628	**3.1574**
128	3.1309	**3.1289**	3.1353	**3.1326**
64	3.1312	**3.1269**	3.1312	**3.1288**
32	3.1354	**3.1251**	3.1299	**3.1269**

factor, we can find that asymmetric representation range even making block 32 quantization worse than block 256 quantization. This is because only the minimal values in each quantization block have possibility to quantized into 128 in INT8 quantization, and smaller block size indicats more individual quantization blocks. Additionally, asymmetric quantization also cause degeneration for UE8M0 scale factors, but the degeneration strength is slighter than BFloat16 scales. This is because UE8M0 scale factor consistently greater than or equal to Bfloat16 scale, leading less high-precision number to map to Q_{min}. These experiments demonstrate the necessity of symmetric representation space for integer quantization.

Algorithm 1 Analyzing Numerical Stability of Different Floating-Point Precisions

Input: Dimension $N = 4096$, precision list $P = \{\text{bfloat16}, \text{float16}, \text{float32}\}$
Output: Ratio of elements equal to 128 for each precision
for each *precision* in P **do**
    $D \leftarrow \text{GenerateRandomMatrix}(N, N, \text{precision})$ ▷ Generate $N \times N$ matrix from $\mathcal{N}(0, 1)$ on GPU
    $S \leftarrow D/127$ ▷ Calculate the scaler matrix
    $D_{\text{norm}} \leftarrow \text{Round}(D \oslash S)$ ▷ $\oslash$ denotes element-wise division
    $count \leftarrow \text{CountElementsEqualTo}(D_{\text{norm}}, 128)$
    $total \leftarrow N \times N$
    $ratio \leftarrow count/total$
    print "Precision:", *precision*, ", Ratio:", $ratio$

Table 9: Results of Algorithm 1.

BFloat16	Float16	Float32
16.82%	0.02%	0

Numerical stability analysis. We also analyze the numerical stability of different float-point for quantization mapping through Algorithm 1. Table 9 shows the results of Algorithm 1, demonstrating that in BFloat16 precision, a significant portion of values (16.82%) are mapped to -128. This phenomenon occurs even though the scaling factor s is theoretically designed to map the value to 127. In conclusion, this analysis highlights a critical pitfall of using low-precision floating-point formats for quantization calculations. The inherent lack of precision in bfloat16 and, to a lesser extent, float16 can lead to overflow during the scaling step, incorrectly mapping values to outside the intended integer range. This powerfully demonstrates that a forced symmetric clipping step is essential for guaranteeing the correctness and stability of quantization, particularly when the computation is performed using low-precision data types.

D.3 Detailed Results

This section offer detailed numbers of experiments, as follows:

- Table 10 and Table 11 present the KL divergence results, corresponding to Figure 3.
- Table 12 and Table 13 present the perplexity results, for better understanding the relationship between KL divergence and perplexity. They are consistent in most case.
- Table 14 and Table 15 presents the item-wise QSNR results, corresponding to Figure 6.

Table 10: Qwen3 models KL divergence (lower is better) results across different low-bit formats in direct-cast inference. All reported KL metrics are the average over all tokens, multiplied by 10^6.

	Qwen-3							
Format	0.6B	1.7B	4B	8B	14B	32B	30B-A3B	235B-A22B
MXINT8	**191**	**209**	**112**	**168**	**96**	**118**	**160**	**276**
MXFP8	624	434	370	380	320	486	400	493
MXINT6	1944	2464	928	1104	804	1012	768	1333
MXFP6	**1136**	**948**	**612**	**636**	**512**	**688**	**536**	**1117**
MXINT4	**39936**	**30208**	**17408**	**15552**	34304	27392	13248	16331
MXFP4	41472	33024	20096	**15744**	**12928**	**13056**	12096	22710
NVINT4	**10560**	**8320**	**4864**	**5120**	5568	7968	**3120**	**9702**
NVFP4	15040	10944	6816	6272	**5536**	**5536**	3936	9979
	Qwen-3 **(w/ random Hadamard rotation)**							
Format	0.6B	1.7B	4B	8B	14B	32B	30B-A3B	235B-A22B
MXINT8	**137**	**150**	**80**	**130**	**70**	**88**	**135**	**229**
MXFP8	937	1366	493	596	424	543	417	818
MXINT6	1137	**1274**	547	690	**481**	**615**	**444**	**809**
MXFP6	**1099**	1549	542	679	500	617	455	810
MXINT4	**26488**	**26578**	**10498**	**12241**	**8459**	**9510**	**6080**	**9660**
MXFP4	48788	45624	15801	18731	12781	11274	10506	12086
NVINT4	**7771**	**7236**	**3431**	**4026**	**3070**	**3647**	**2222**	**3931**
NVFP4	18002	18761	7753	8329	6372	6822	5605	7790

Table 11: Llama-3 models KL divergence (lower is better) results across different low-bit formats in direct-cast inference. All reported KL metrics are the average over all tokens, multiplied by 10^6.

	Llama			
Format	3.2-1B	3.2-3B	3.1-8b	3.1-70B
MXINT8	**111**	**77**	**82**	**191**
MXFP8	504	358	401	548
MXINT6	1133	743	776	1744
MXFP6	**754**	**520**	**569**	**1499**
MXINT4	**26153**	**14089**	**12380**	**22538**
MXFP4	42896	27586	40015	41396
NVINT4	**7508**	**4312**	**4224**	**10970**
NVFP4	14048	8590	8356	12929
	Llama(**w/ random Hadamard rotation**)			
Format	3.2-1B	3.2-3B	3.1-8b	3.1-70B
MXINT8	**89**	**63**	**65**	**145**
MXFP8	632	429	445	1945
MXINT6	773	531	558	1518
MXFP6	**742**	**511**	**530**	**2984**
MXINT4	**20126**	**11116**	**10272**	**137612**
MXFP4	34884	28449	27023	171170
NVINT4	**5854**	**3912**	**3609**	**19975**
NVFP4	15436	9950	3611	112772

Table 12: Qwen3 models perplexity (lower is better) results of WikiText2 across different low-bit formats in direct-cast inference.

	Qwen-3							
Format	0.6B	1.7B	4B	8B	14B	32B	30B-A3B	235B-A22B
Bf16	11.5868	8.7084	7.3368	6.5135	5.9498	7.0168	6.8178	4.0929
MXINT8	**11.6377**	**8.7424**	**7.3511**	**6.5174**	**5.955**	**7.0185**	**6.8167**	**4.0959**
MXFP8	11.762	8.7873	7.3823	6.5465	5.971	7.0392	6.8407	4.1116
MXINT6	12.2297	9.2622	7.496	6.6499	6.0483	7.05	6.8745	4.1743
MXFP6	**11.9379**	**8.9082**	**7.4275**	**6.5816**	**5.9913**	**7.0405**	**6.8449**	**4.1639**
MXINT4	**48.6713**	**21.8749**	**11.9487**	10.0423	16.7227	15.1619	9.3837	**5.918**
MXFP4	45.9304	24.0766	12.4515	**9.6166**	**8.04**	**8.577**	**9.1905**	7.0814
NVINT4	**15.9729**	**10.9128**	**8.3304**	**7.415**	6.81	8.0161	**7.2024**	**4.8916**
NVFP4	17.8613	11.7405	8.6055	7.4821	**6.642**	**7.5168**	7.5167	4.907
	Qwen-3(**w/ random Hadamard rotation**)							
Format	0.6B	1.7B	4B	8B	14B	32B	30B-A3B	235B-A22B
MXINT8	**11.6179**	**8.7240**	**7.3407**	**6.5170**	**5.9521**	**7.0187**	**6.8231**	**4.0973**
MXFP8	11.8935	9.0039	7.4136	6.5896	5.9892	7.0709	6.8849	4.1308
MXINT6	**11.9422**	**9.0122**	**7.4071**	6.6119	5.9905	**7.0627**	**6.8666**	**4.1263**
MXFP6	11.9491	9.0448	7.4159	**6.5805**	**5.9887**	7.0746	6.8670	4.1391
MXINT4	**28.6510**	**21.3032**	**9.8238**	**9.2029**	**7.3564**	**8.2083**	**7.8292**	**4.9891**
MXFP4	54.6892	35.1683	11.1139	10.5028	8.0017	8.5446	9.1705	5.2541
NVINT4	**14.6052**	**10.7822**	**7.9824**	**7.1705**	**6.3702**	**7.3625**	**7.1557**	**4.3913**
NVFP4	20.6018	15.1028	8.9165	7.9712	6.8207	7.8472	8.0406	4.8161

Table 13: Llama-3 models perplexity (lower is better) results of WikiText2 across different low-bit formats in direct-cast inference.

	Llama			
Format	3.2-1B	3.2-3B	3.1-8b	3.1-70B
BF16	**9.0625**	**7.2857**	**5.8402**	**2.637**
MXINT8	**9.0815**	**7.2944**	**5.8487**	**2.6449**
MXFP8	9.1787	7.3427	5.9014	2.6688
MXINT6	9.3557	7.4184	5.9643	2.7493
MXFP6	**9.2434**	**7.3685**	**5.9264**	**2.7329**
MXINT4	**21.9893**	**11.2715**	**8.7408**	**5.1894**
MXFP4	33.1409	16.4374	20.792	10.0413
NVINT4	**11.3987**	**8.225**	**6.5957**	**3.5502**
NVFP4	13.6606	9.1858	7.4047	3.7188
	Llama(**w/ random Hadamard rotation**)			
Format	3.2-1B	3.2-3B	3.1-8b	3.1-70B
MXINT8	**9.0715**	**7.2912**	**5.845**	**2.6428**
MXFP8	9.2127	7.3526	5.9109	2.7673
MXINT6	9.2622	7.3828	5.9276	2.7333
MXFP6	**9.2502**	**7.372**	**5.9234**	**2.8547**
MXINT4	**17.9797**	**10.3057**	**8.0745**	**1146.7256**
MXFP4	26.6788	17.1619	13.2289	5600.7686
NVINT4	**10.8399**	**8.1119**	**6.4701**	**4.9786**
NVFP4	14.5804	9.6662	6.4932	216.3876

Table 14: QSNR versus block size for quantization across 6 types of forward/backward items. -1 denotes per-channel quantization. E4M3 scales contains another FP32 per-tensor scale factor just like NV-format.

Format	Scales Type	Item	G = -1	256	128	64	32	16
INT8	E8M0	① $\mathbf{X}$	27.55	33.73	35.41	37.20	38.96	40.74
		② $\mathbf{W}$	36.81	38.97	39.72	40.59	41.51	42.47
		③ $\mathbf{dY}$	29.00	35.02	36.60	37.87	39.30	40.50
		④ $\mathbf{W}^T$	36.00	38.69	39.56	40.42	41.45	42.44
		⑤ $\mathbf{X}^T$	36.46	38.71	39.59	40.54	41.56	42.65
		⑥ $\mathbf{dY}$	25.12	31.09	33.11	35.08	37.04	38.85
FP8	E8M0	① $\mathbf{X}$	31.21	31.23	31.23	31.23	31.23	31.23
		② $\mathbf{W}$	31.27	31.27	31.27	31.27	31.27	31.27
		③ $\mathbf{dY}$	31.22	31.23	31.23	31.23	31.23	31.22
		④ $\mathbf{W}^T$	31.27	31.27	31.27	31.27	31.27	31.27
		⑤ $\mathbf{X}^T$	31.23	31.23	31.23	31.23	31.23	31.23
		⑥ $\mathbf{dY}$	31.22	31.22	31.22	31.22	31.22	31.22
INT6	E8M0	① $\mathbf{X}$	15.69	21.66	23.33	25.15	26.96	28.82
		② $\mathbf{W}$	24.68	26.91	27.68	28.58	29.60	30.70
		③ $\mathbf{dY}$	17.84	23.30	24.82	26.19	27.83	29.41
		④ $\mathbf{W}^T$	23.87	26.62	27.51	28.41	29.53	30.66
		⑤ $\mathbf{X}^T$	24.52	26.74	27.62	28.61	29.69	30.87
		⑥ $\mathbf{dY}$	15.68	20.95	22.82	24.70	26.70	28.74
FP6	E8M0	① $\mathbf{X}$	20.78	26.02	27.22	28.33	29.21	29.89
		② $\mathbf{W}$	28.42	29.58	29.84	30.10	30.34	30.55
		③ $\mathbf{dY}$	22.20	27.02	28.05	28.81	29.55	30.08
		④ $\mathbf{W}^T$	27.89	29.43	29.77	30.04	30.32	30.54
		⑤ $\mathbf{X}^T$	27.91	29.20	29.59	29.95	30.26	30.52
		⑥ $\mathbf{dY}$	19.60	24.63	26.15	27.49	28.65	29.54
INT4	E4M3	① $\mathbf{X}$	7.33	12.52	14.13	15.91	17.70	19.59
		② $\mathbf{W}$	14.71	17.41	18.19	19.03	19.98	21.12
		③ $\mathbf{dY}$	9.72	14.11	15.40	16.74	18.33	19.98
		④ $\mathbf{W}^T$	14.01	17.08	17.99	18.84	19.91	21.08
		⑤ $\mathbf{X}^T$	15.26	17.42	18.27	19.21	20.28	21.51
		⑥ $\mathbf{dY}$	9.56	13.73	15.33	16.96	18.76	20.69
FP4	E4M3	① $\mathbf{X}$	9.88	14.17	15.32	16.53	17.70	18.96
		② $\mathbf{W}$	15.33	16.60	16.94	17.33	17.85	18.58
		③ $\mathbf{dY}$	11.43	15.00	15.85	16.76	17.79	18.94
		④ $\mathbf{W}^T$	14.89	16.47	16.87	17.29	17.84	18.58
		⑤ $\mathbf{X}^T$	15.42	16.53	16.93	17.38	17.93	18.64
		⑥ $\mathbf{dY}$	11.43	15.42	16.80	18.17	19.62	21.19
INT4	E8M0	① $\mathbf{X}$	5.25	9.71	11.18	12.85	14.53	16.29
		② $\mathbf{W}$	11.86	14.18	14.82	15.60	16.57	17.70
		③ $\mathbf{dY}$	7.57	11.46	12.64	13.90	15.41	16.96
		④ $\mathbf{W}^T$	11.28	13.85	14.65	15.44	16.50	17.66
		⑤ $\mathbf{X}^T$	12.29	14.24	15.02	15.92	16.93	18.07
		⑥ $\mathbf{dY}$	8.11	11.72	13.12	14.55	16.11	17.77
FP4	E8M0	① $\mathbf{X}$	7.62	11.86	12.99	14.13	15.11	15.94
		② $\mathbf{W}$	13.11	15.13	15.48	15.82	16.17	16.55
		③ $\mathbf{dY}$	9.65	13.11	13.94	14.73	15.55	16.21
		④ $\mathbf{W}^T$	12.74	14.90	15.36	15.73	16.14	16.54
		⑤ $\mathbf{X}^T$	13.55	14.96	15.41	15.86	16.29	16.69
		⑥ $\mathbf{dY}$	9.55	12.72	13.74	14.66	15.51	16.22

Table 15: QSNR versus block size **with random Hadamard rotation** for quantization across 6 types of forward/backward items. -1 denotes per-channel quantization. E4M3 scales contains another FP32 per-tensor scale factor just like NV-format.

Format	Scale Type	Item	G = -1	256	128	64	32	16
INT8	E8M0	① $\mathbf{X}$	35.07	39.32	40.30	41.24	42.27	43.06
		② $\mathbf{W}$	37.95	39.46	40.12	40.89	41.75	42.63
		③ $\mathbf{dY}$	34.55	39.02	40.27	41.07	42.01	42.47
		④ $\mathbf{W}^T$	37.56	39.23	39.99	40.76	41.74	42.63
		⑤ $\mathbf{X}^T$	36.76	39.05	39.92	40.83	41.82	42.66
		⑥ $\mathbf{dY}$	32.94	38.63	40.39	42.03	43.51	43.67
FP8	E8M0	① $\mathbf{X}$	31.26	31.26	31.26	31.26	31.26	31.26
		② $\mathbf{W}$	31.25	31.25	31.25	31.25	31.25	31.25
		③ $\mathbf{dY}$	31.24	31.24	31.24	31.24	31.24	31.23
		④ $\mathbf{W}^T$	31.25	31.25	31.25	31.25	31.25	31.25
		⑤ $\mathbf{X}^T$	31.26	31.26	31.26	31.26	31.26	31.26
		⑥ $\mathbf{dY}$	31.24	31.24	31.24	31.24	31.24	31.24
INT6	E8M0	① $\mathbf{X}$	22.93	27.24	28.28	29.28	30.40	31.31
		② $\mathbf{W}$	25.86	27.43	28.09	28.91	29.86	30.87
		③ $\mathbf{dY}$	22.69	27.10	28.36	29.32	30.55	31.41
		④ $\mathbf{W}^T$	25.45	27.20	27.96	28.77	29.85	30.87
		⑤ $\mathbf{X}^T$	24.60	26.94	27.90	28.85	29.90	30.86
		⑥ $\mathbf{dY}$	21.32	26.95	28.80	30.64	32.47	33.03
FP6	E8M0	① $\mathbf{X}$	27.15	29.65	30.03	30.31	30.54	30.67
		② $\mathbf{W}$	29.13	29.80	29.99	30.20	30.40	30.58
		③ $\mathbf{dY}$	26.72	29.51	30.02	30.29	30.55	30.67
		④ $\mathbf{W}^T$	28.87	29.70	29.95	30.16	30.40	30.58
		⑤ $\mathbf{X}^T$	28.29	29.55	29.87	30.18	30.43	30.60
		⑥ $\mathbf{dY}$	25.66	29.21	29.92	30.43	30.78	30.84
INT4	E4M3	① $\mathbf{X}$	12.64	17.12	18.28	19.44	20.65	22.02
		② $\mathbf{W}$	15.72	17.91	18.59	19.33	20.22	21.30
		③ $\mathbf{dY}$	12.80	16.99	18.22	19.24	20.54	21.92
		④ $\mathbf{W}^T$	15.33	17.60	18.41	19.18	20.18	21.30
		⑤ $\mathbf{X}^T$	14.73	17.11	18.06	19.08	20.19	21.38
		⑥ $\mathbf{dY}$	11.99	16.86	18.55	20.28	22.07	23.92
FP4	E4M3	① $\mathbf{X}$	13.95	16.51	17.00	17.45	17.95	18.56
		② $\mathbf{W}$	15.86	16.77	17.05	17.39	17.87	18.56
		③ $\mathbf{dY}$	13.95	16.45	16.96	17.39	17.92	18.56
		④ $\mathbf{W}^T$	15.65	16.66	17.00	17.36	17.87	18.56
		⑤ $\mathbf{X}^T$	15.31	16.53	16.93	17.39	17.93	18.62
		⑥ $\mathbf{dY}$	13.77	17.00	17.81	18.49	19.07	19.57
INT4	E8M0	① $\mathbf{X}$	9.64	13.85	14.97	16.06	17.21	18.50
		② $\mathbf{W}$	12.89	14.65	15.20	15.89	16.80	17.88
		③ $\mathbf{dY}$	10.17	14.04	15.18	16.18	17.44	18.75
		④ $\mathbf{W}^T$	12.45	14.39	15.06	15.75	16.76	17.88
		⑤ $\mathbf{X}^T$	11.47	13.84	14.71	15.71	16.73	17.89
		⑥ $\mathbf{dY}$	9.92	14.22	15.77	17.33	18.96	20.64
FP4	E8M0	① $\mathbf{X}$	11.78	14.75	15.41	15.96	16.42	16.84
		② $\mathbf{W}$	13.74	15.40	15.69	15.97	16.28	16.63
		③ $\mathbf{dY}$	12.10	14.90	15.52	15.99	16.49	16.90
		④ $\mathbf{W}^T$	13.58	15.22	15.61	15.90	16.27	16.63
		⑤ $\mathbf{X}^T$	13.16	14.74	15.25	15.75	16.20	16.63
		⑥ $\mathbf{dY}$	11.67	14.74	15.57	16.27	16.85	17.29