An Empirical Study of Microscaling Formats for Low-Precision LLM Training

Hanmei Yang*†, Summer Deng*, Amit Nagpal*, Maxim Naumov*, Mohammad Janani*, Tongping Liu†, Hui Guan†,
*Meta, †University of Massachusetts Amherst
Email: {hanmei, summerdeng, amitnagpal, mnaumov, mjanani}@meta.com, {tongping, huiguan}@umass.edu

***Abstract*—This paper presents a comprehensive evaluation of microscaling (MX) quantization in the pre-training of large language models (LLMs), investigating its potential to enhance the computation and memory efficiencies. We systematically examine the effects of key design parameters - including data types, rounding modes, scaling strategies, granularity, and organization - on numerical accuracy and training stability. Our extensive experimental study on Llama3 models reveals critical insights into the challenges of 4-bit training for LLMs and identifies optimal configurations with mixed precisions of 4-bit and 6-bit MX formats that significantly enhance training quality, bridging the gap with higher-precision formats. This research provides valuable guidance on the benefits and limitations of MX quantization, laying the groundwork for future innovations in low-precision LLM training.**

I. Introduction

The recent rapid progress in natural language processing has been driven by remarkable advancements in large language models (LLMs), which achieve state-of-the-art performance in tasks such as machine translation, conversational agents, and text summarization. However, their impressive capabilities come with significant computational and memory demands, posing substantial challenges for efficient training and inference. To address these challenges, quantization techniques have gained prominence, reducing numerical precision to improve efficiency. Formats like FP16 [8], BF16 [7], FP8 [6], [9], and INT8 [5] have demonstrated notable success, but it remains uncertain whether precision can be further reduced to 4 bits without compromising convergence or model quality.

Microscaling (MX) formats offer a promising direction [4], [12], employing shared scaling factors across small groups of elements to reduce precision while preserving a wide numerical range. The OCP specification [11] defines the following MX formats: 1) MXFP8, which supports both E5M2 and E4M3 data types, where E represents exponent bitwidth and M represents mantissa bitwidth; 2) MXFP6, which includes E3M2 and E2M3; 3) MXFP4, which utilizes the E2M1 data type; and 4) MXINT8, which uses an INT8 data type. All the OCP MX formats use a group size of 32 and the E8M0 format for the shared scales. Despite their success in inference tasks [12], the application to LLM training remains largely unexplored, leaving a significant knowledge gap.

This paper bridges this gap by conducting the first comprehensive study of MX quantization in LLM pre-training. We systematically investigate the numerical fidelity of MX formats during training, exploring their design space to identify new recipes that enable stable and efficient training. Our research focuses on two questions: (1) How does the default MX quantization method, as specified in the OCP MX specification [11], impact numerical fidelity during LLM pre-training? (2) How can we optimize MX quantization to achieve better stability and performance? To answer these questions, we performed extensive experiments on 7B-scale Llama [14] models, evaluating the numerical fidelity of MX quantization in various configurations. Key contributions of this work include:

- A systematic study of MX quantization for LLM training, examining key design choices and their impact on numerical fidelity and training stability.
- Highlighting the critical insights on the design choices and identifying effective configurations that leverage a mix of 4-bit and 6-bit precision to enhance the viability of low-precision LLM training.
- Demonstrating that carefully tailored MX configurations can match the training loss performance of FP8 and BF16 baselines, providing valuable guidance to achieve high fidelity low-precision LLM training.

II. Training with MX Formats

A. Training Workflow

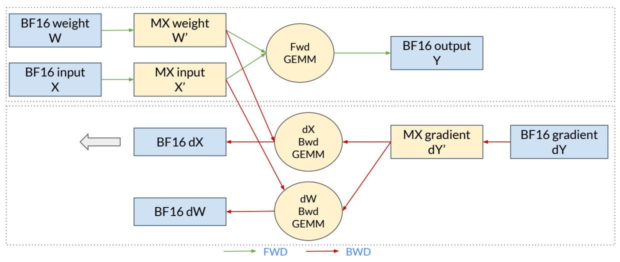


Fig. 1: **MX Quantization Emulation Workflow**. Weights, activations, and gradients are quantized into the MX format during forward and backward passes. As current hardware lacks native support for MX computation, the quantized MX tensors are cast back to BF16 for GEMM operations.

Figure 1 illustrates the MX quantization workflow applied to all linear modules in the Llama model, including QKV and output projections in the attention layers, as well as the fully connected layers within the MLPs. All other layers operate at the default BF16 precision. To ensure numerical stability, our workflow employs mixed-precision training with a high-precision weight copy (FP32 or BF16) for the optimizer, while its memory overhead is mitigated by distributing model parameters across GPUs with FSDP [16]. As the existing hardware does not support MX formats yet, we emulated them with fake quantization. Namely, in the forward pass, weights and activations are first quantized into the MX format, then cast back to BF16, before the BF16 GEMM computation. Similarly, in the backward pass, gradients are fake-quantized into the MX format and then perform BF16 GEMM computation. This emulation is hardware-agnostic and implemented with the torchao library [13]. It enables preliminary numerical investigations of MX formats on existing hardware, a widely used approach in prior work [4], [12], before native MX hardware became available. Recently, with access to NVIDIA B200 GPUs, we validated our emulation by comparing GEMM outputs against real MX computations and confirmed that the emulated GEMM produces identical results.

B. Performance Analysis

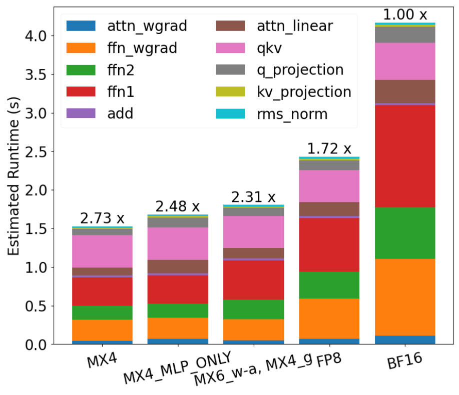


Fig. 2: Runtime breakdown of computation for the Llama3 405B model across different quantization configurations, with speedup factors relative to the BF16 baseline.

To evaluate the potential performance benefits of MX formats, we utilized a performance projection framework that simulates end-to-end LLM training performance based on empirical measurement data on today's hardware. Since MX formats are not natively supported, we estimated the MXFP4 and MXFP6 GEMM kernel performance by extrapolating the performance gains observed when transitioning from BF16 to FP8 precision on different tensor shapes. While these projections offer an informative reference, the actual performance ultimately depends on vendor-specific hardware implementations. For example, if MXFP6 achieves the same throughput as MXFP8, as seen in Blackwell GPUs [10], MX6_w-a MX4_g can achieve a $2.11\times$ speedup; if it aligns with MXFP4, the speedup increases to $2.4\times$. Figure 2 presents the estimated runtime breakdown for the Llama3 405B model across various quantization configurations. Notable observations include:

- MX4 achieves the highest speedup of $2.73\times$, showing the maximum efficiency of 4-bit quantization.
- MX4_MLP_ONLY applies MX quantization only to the linear layers in the MLP module but not the attention module. This option delivers a $2.48\times$ speedup, slightly less than MX4 for all linear layers, indicating that MLP layers are the primary computational bottleneck.
- MX6_w-a, MX4_g strikes a balance between precision and performance with a $2.31\times$ speedup, by using MXFP6 on weights and activations, while MXFP4 on gradients.
- FP8, which employs tensor-wise scaling factors, delivers a $1.72\times$ speedup relative to the BF16 and serves as a baseline for evaluating MX quantization configurations.

Beyond runtime efficiency, MX formats can also provide substantial communication savings with reduced message sizes. In MX formats, each block shares a common 8-bit scaling factor (E8M0), which just adds an extra of $8/block\_size$ bits to the element bitwidth. For example, MXFP4 with a block size of 32 has 4.25 bits per element, achieving a $3.76\times$ reduction in memory and communication overhead compared to BF16. The benefits of reduced data transfer are particularly significant in large-scale model training, leading to improved inter-GPU communications and scalability. Although this paper focuses on runtime analysis, a detailed investigation of communication savings remains a topic for further research.

III. Design Choices for MX Quantization

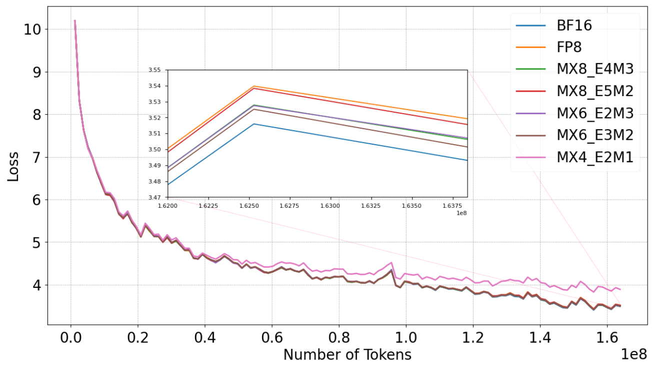


Fig. 3: Training loss curves for MXFP4, MXFP6, MXFP8, FP8, and BF16 baseline on the Llama3 7B model, with an inset highlighting final loss comparisons.

Figure 3 presents the training loss curves in terms of cross-entropy loss for the Llama3 7B model under various quantization configurations, including MXFP4 (E2M1), MXFP6 (E2M3, E3M2), MXFP8 (E4M3, E5M2), and baseline formats (FP8, BF16), over 160 million tokens. Among these configurations, MX4_E2M1 diverges early after processing around 40 million tokens, while the others appear nearly identical in the main plot, hence the inset provides a more detailed comparison. Specifically, MXFP6 and MXFP8 closely match the FP8 and BF16 baselines, with loss differences within 0.025. In contrast, MXFP4 exhibits a notably larger loss gap of up to 0.5 compared to BF16 and continues to grow, indicating that the gap will widen with further training. This growing disparity uncovers the challenges of default 4-bit MX quantization and motivates a systematical exploration of the

MX design space to overcome these limitations. In this section, we outline the key design factors driving our investigations.

A. Data Types

Data types define the numerical formats used to represent each element in a block. Common data types for 4-bit quantization are as follows:

- **E2M1**: The default MXFP4 format as defined in the OCP specification, consists of 1 sign bit, 2 exponent bits, and 1 mantissa bit.
- **E3M0**: A format with 1 sign bit, 3 exponent bits, and no mantissa, providing maximum dynamic range but reduced precision. It is recommended for gradients in [2], where the data is better represented with lognormal distributions.
- **INT4**: A 4-bit integer format that offers better precision but with limited dynamic range.

Each data type has unique trade-offs between dynamic range and precision, making specific formats more compatible with certain tensor characteristics. Choosing the optimal configuration is crucial for adapting quantization strategies to the data distributions, thus leading to enhanced training accuracy.

B. Element Rounding Modes

Element rounding mode specifies how each value is rounded to the nearest representable value during conversion to low precision. Commonly used rounding modes include:

- **Round to Nearest, Ties to Even (RTNE)**: Rounds each value to the closest representable number, breaking ties by rounding to the nearest even value. As the default rounding mode in the OCP specification, RTNE effectively reduces rounding bias in aggregate computations and helps maintain numerical stability during training.
- **Stochastic Rounding (SR)** [3]: Unlike deterministic rounding, SR rounds a value up or down with a probability proportional to its proximity to the nearest representable values. This method provides a better chance to preserve the small-magnitude values instead of rounding to 0, making it particularly effective for gradient quantization in low-precision training. For a real number x between two representable values v_{low} and v_{high}, the rounded value $\text{Round}(x)$ is determined as follows:

$$\text{Round}(x) = \begin{cases} v_{\text{low}}, & \text{if } P < \frac{v_{\text{high}} - x}{v_{\text{high}} - v_{\text{low}}}, \\ v_{\text{high}}, & \text{if } P \geq \frac{v_{\text{high}} - x}{v_{\text{high}} - v_{\text{low}}}, \end{cases} \tag{1}$$

where $P \sim \mathcal{U}(0, 1)$ is a random variable sampled from a uniform distribution.

Other rounding modes, such as round ties away, exist but are not suitable for training due to rounding biases or larger errors. Therefore, they are not considered in this work.

C. Scale Rounding Modes

The OCP specification defines the scaling format in E8M0, which restricts values to power-of-2 floating points. Hence, a rounding mode is required to convert the maximum element value to the nearest power-of-2, which will be used in the following down-conversion of the element values. This choice significantly impacts the dynamic range and precision of elements. We examined three rounding strategies in this work:

- **Floor**: Recommended by the OCP specification, this method calculates the scale S by selecting the maximum absolute value $(\max(|V_i|))$ within a group of elements, taking its base-2 logarithm, subtracting the maximum exponent (maxExp) specific to the data type, and rounding down to the nearest integer $(\lfloor \cdot \rfloor)$:

$$S = 2^{\lfloor \log_2(\max(|V_i|)) - \text{maxExp} \rfloor}, \tag{2}$$

where maxExp is a constant specific to the element data type (e.g., 2 for E2M1 and 4 for E3M0). While this method is hardware-efficient, its truncation-based scaling often causes overflows during element down-conversion in low-precision formats such as MXFP4. This leads to large values being clamped to the maximum representable value, which introduces substantial quantization errors and notable degradation in training accuracy.

- **Ceil**: Instead of truncating, this method rounds up the logarithm of the maximum absolute value to the nearest integer $(\lceil \cdot \rceil)$ after subtracting the maximum exponent. The scale S is calculated as:

$$S = 2^{\lceil \log_2(\max(|V_i|)) - \text{maxExp} \rceil}. \tag{3}$$

Unlike **Floor**, which risks at clamping large magnitudes to the maximum representable value, **Ceil** avoids this issue by selecting the smallest power-of-2 that is equal to or greater than $\max(|V_i|)$. This ensures that elements are covered in the representable range. Although it eliminates the overflow issues, the increased shared scale could shift more small-magnitude values to zero, exacerbating underflow issues for smaller elements.

- **Even**: We propose a new method that uses RTNE mode for scale calculation, which rounds the maximum absolute value at the bit position set by the target mantissa bit width before computing the logarithm:

$$S = 2^{\lfloor \log_2(\text{Round}(\max(|V_i|))) - \text{maxExp} \rfloor}. \tag{4}$$

Instead of consistently rounding in one direction, **Even** determines the scale based on the numerical distribution, reducing systematic bias in scale selection. By adaptively balancing **Floor** and **Ceil**, it mitigates excessive clamping of large magnitudes due to limited mantissa precision while better preserving smaller ones.

D. Symmetric vs. Asymmetric Scaling

Symmetric and asymmetric scaling define how the scaling factor captures the range and shift of data distributions:

- **Symmetric Scaling:** The scaling factor is derived from the absolute maximum value within a block, as shown in Equations 2–4. This ensures a symmetric range centered around zero, simplifying hardware implementation.

- **Asymmetric Scaling:** To better adapt to data distributions not centered at zero, this method introduces an offset to shift the quantization range. Specifically, the scaling factor is determined by the range between the maximum and minimum values within a block, and the offset is calculated as $O = \text{FP16}\left(\frac{\max(V_i)+\min(V_i)}{2}\right)$, where $\text{FP16}(\cdot)$ denotes conversion to half-precision floating point.

Asymmetric scaling better accommodates skewed data distributions, making more efficient use of the quantization range. However, introducing an offset incurs additional arithmetic operations, which are typically not compute-intensive due to vector processing but tend to be memory-bound. Dedicated hardware can help mitigate these costs, but integrating offset computations into the pipeline adds logic overhead and design complexity. As a result, the practicality of asymmetric scaling depends on hardware resources and the trade-off between precision and implementation overhead.

E. Scaling Granularity and Organization

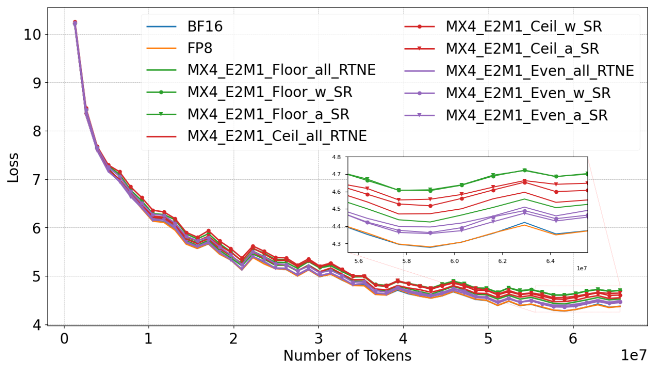
Scaling granularity determines the number of elements sharing a scaling factor within a block, and the organization specifies their spatial arrangement. The effectiveness of these configurations heavily depends on the underlying data distribution. Configurations tailored to match data patterns can optimize precision and system efficiency, whereas arbitrary choices can result in inefficiencies and compromised accuracy.

- **Granularity**: The block size choice in MX formats involves a trade-off between precision and efficiency. Smaller blocks capture the data variations more precisely at a finer granularity, but at the cost of increased storage and computational overhead. Larger blocks, on the other hand, reduce the storage and computation cost but may compromise precision, particularly for tensors with outliers or multimodal data distributions.
- **Organization**: Organization defines how the blocks are arranged within a tensor. The default MX formats use **row-wise** scaling, where blocks of elements are sequentially selected along the row dimension. Canonically, the row dimension refers to the reduction dimension in GEMM operators as the scalar scaling factors along the reduction dimension pose little overhead in the GEMM kernel. Other layout options include **column-wise** scaling, where blocks are formed along the column dimension, and **2D block-wise** scaling, which applies scaling factors to rectangular tensor sections. These alternative layouts can more effectively capture feature patterns in the data, but necessitate additional layout transformations, adding extra overhead to GEMM operations.

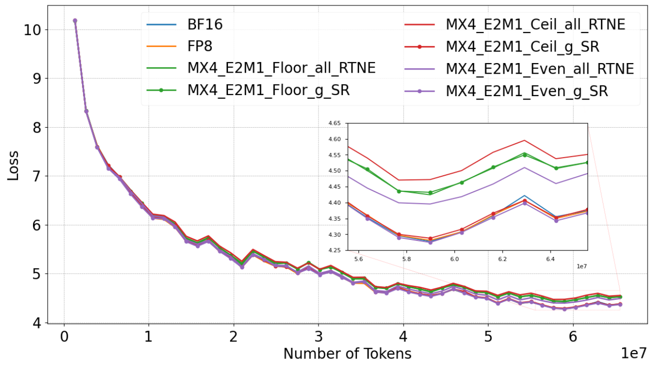
IV. Experiment Results

A. Effectiveness of Each Design Choice

Figure 4 shows the impact of scale rounding modes (Floor, Ceil, Even) and element rounding modes (RTNE, SR) on the training loss of the Llama3 7B model with MX4_E2M1 quantization. RTNE is applied uniformly across all tensor types,



(a) SR is applied to weights (w) and activations (a) in the forward pass.



(b) SR is applied to gradients (g) in the backward pass.

Fig. 4: Impact of scale rounding modes (Floor, Ceil, Even) and element rounding modes (RTNE, SR) on training loss for the Llama3 7B model under MX4_E2M1 quantization. The results are compared against the FP8 and BF16 baselines.

while SR is analyzed separately for weights and activations in Figure 4a, and for gradients in Figure 4b. The results are compared to MX4_E2M1_{Floor, Ceil, Even}_all_RTNE.

From Figure 4a, when RTNE is applied to all tensors, Floor performs poorly due to overflow issues. Ceil, while avoiding overflow, exacerbates underflow problems, which results in suboptimal performance. In contrast, Even outperforms Floor and Ceil by balancing both overflow and underflow effects. When Stochastic Rounding (SR) is applied to weights and activations, the performance of Floor and Ceil degrades significantly due to the larger quantization errors introduced by SR. Among these, Floor performs the worst because SR amplifies overflow issues, which impose higher penalties compared to errors caused by small-magnitude values. Overall, these findings suggest that applying SR to weights and activations generally hurts training stability and accuracy.

In contrast, Figure 4b demonstrates a distinct trend when SR is applied to gradients. Both Ceil and Even rounding modes show significant improvements with SR, achieving training losses comparable to the FP8 and BF16 baselines. This result highlights the critical role of SR in the gradient quantization, which is consistent with previous findings [1], [15]. Underflow issues in gradient quantization often cause small values to be rounded to zero. As training progresses, this rounding effect accumulates, leading to a gradual degradation of gradients that severely compromises training stability. Although Floor rounding mode slightly mitigates underflow issues when combined with SR, its high susceptibility to overflow penalties results in insufficient accuracies. In summary, combining SR with Ceil or Even rounding yields competitive results. In subsequent experiments, RTNE is applied to weights and activations, while SR is used for gradients.

Insight 1: For element rounding mode, RTNE is effective for weight and activation quantization, whereas SR is better suited for quantizing gradients.

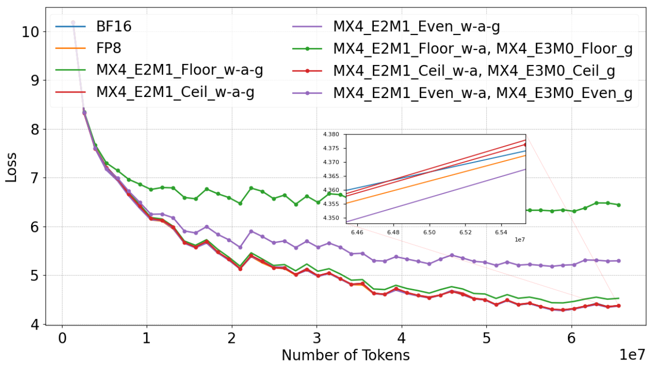


Fig. 5: Impact of E3M0 on gradient quantization, comparing E2M1 and E3M0 formats for gradients in the Llama3 7B model using 4-bit MX quantization.

Figure 5 examines the impact of using E2M1 and E3M0 formats for gradient quantization in the Llama3 7B model. E3M0 is often regarded as suited for gradient quantization due to the lognormal distribution of gradient magnitudes, which exhibit heavy tails and demand a wide dynamic range [2]. However, despite this theoretical alignment, the results show that E3M0 significantly degrades training performance compared to E2M1 under Floor and Even rounding modes. This performance drop can be attributed to the increased overflow penalties introduced by E3M0's larger dynamic range, which exacerbates the impact of outliers in these rounding modes. For instance, under Even rounding, the overflow rate increases from 0.9% for E2M1 to 3.69% for E3M0, resulting in higher quantization errors. In contrast, Ceil rounding avoids overflow absolutely, enabling both E2M1 and E3M0 to perform comparably to the FP8 and BF16 baselines. These findings emphasize the importance of selecting appropriate rounding modes to mitigate the specific limitations of different data types.

Insight 2: For gradient quantization, E3M0 performs well only with Ceil rounding, while E2M1 works effectively with both Ceil and Even rounding modes.

To efficiently assess quantization effects without conducting full training experiments, we employ rooted mean squared error (R-MSE) as a proxy metric. R-MSE measures the deviation between original and quantized values, providing an efficient and reliable indicator of numerical fidelity. To validate this approach, we analyzed its correlation with training loss gaps by calculating R-MSE from weight tensors at a fixed training step and measuring loss gaps while fixing activation and gradient quantization and varying weight quantization. Figure 6 demonstrates a strong correlation between normalized R-MSE and training loss gap, confirming R-MSE as a viable alternative for evaluating quantization impact. Building on this validation, we employ the R-MSE metric as the proxy to systematically investigate the MX quantization design space in subsequent experiments.

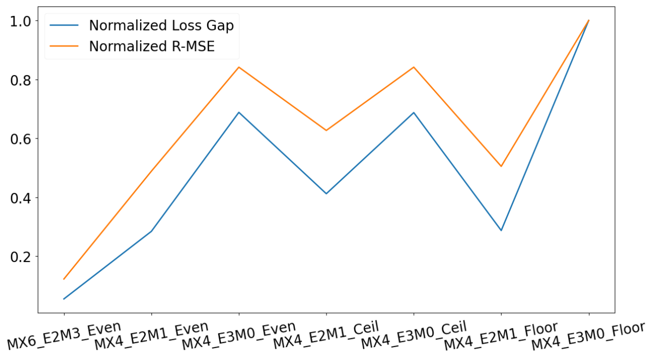


Fig. 6: Correlation between normalized R-MSE and training loss gap for the Llama3 7B model.

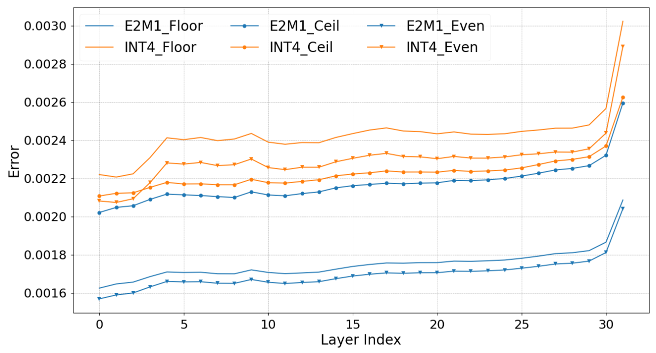


Fig. 7: Impact of scale rounding modes (Floor, Ceil, Even) on R-MSE for the first weight tensor of all MLP layers, comparing E2M1 and INT4 formats.

Figure 7 investigates the impact of scale rounding modes (Floor, Ceil, Even) on R-MSE for the first weight tensor across MLP layers, comparing E2M1 and INT4 formats. The results show that E2M1 consistently outperforms INT4 in terms of R-MSE across all configurations. Specifically, Even rounding mode yields the best performance for E2M1, consistent with previous findings. In contrast, Ceil rounding mode excels for INT4, as its ability to prevent overflow aligns well with INT4's limited representable range. These trends are not limited to weight tensors; similar behaviors are observed for activations and gradients, indicating the general applicability of these findings across different tensor types.

Insight 3: With the E8M0 scale, E2M1 performs best with Even rounding, while INT4 excels with Ceil rounding. Under default MX conditions, E2M1 consistently outperforms INT4.

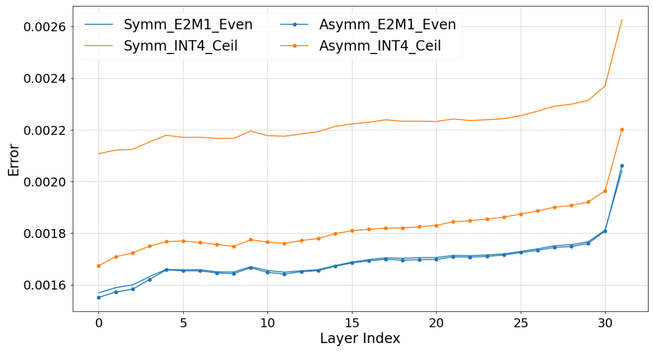


Fig. 8: Impact of symmetric and asymmetric scaling on the R-MSE of the first weight tensor across all MLP layers for E2M1_Even and INT4_Ceil formats.

Figure 8 evaluates the impact of symmetric and asymmetric scaling on the R-MSE of the first weight tensor across all MLP layers for E2M1_Even and INT4_Ceil formats. Asymmetric scaling adjusts the quantization range to better align with skewed distributions, helping to mitigate overflow and underflow errors. For E2M1 with Even rounding, asymmetric scaling slightly reduces R-MSE compared to symmetric scaling, while for INT4 with Ceil rounding, asymmetric scaling significantly reduces R-MSE across all layers. This difference can be attributed to the limited dynamic range and even bit allocation in INT4, which makes it more restrictive to data distributions. Focusing the quantization range on dominant values, asymmetric scaling reduces quantization errors and minimizes the influence of outliers. Similar trends are observed for activations and gradients, as well as for tensors extracted at different training stages. These consistent findings across various tensor types and training phases reinforce the conclusion that the benefits of asymmetric scaling are format-dependent.

Insight 4: Asymmetric scaling significantly improves quantization accuracy for INT4, while offering only slight improvements for E2M1.

Figure 9 illustrates the impact of reducing block size from 32 to 16 on the R-MSE of the first weight tensor across all MLP layers for E2M1_Even and INT4_Ceil formats under symmetric and asymmetric scaling. Reducing the block size generally improves quantization accuracy by capturing finer variations within smaller groups of data. For E2M1_Even with symmetric scaling (blue lines), this results in only slight reductions in R-MSE, as E2M1's higher dynamic range already handles tensor variations well, even with larger block sizes. Asymmetric scaling (green lines) provides a modest additional benefit, but the overall gains remain limited. For INT4_Ceil, block size reduction has a much greater impact.

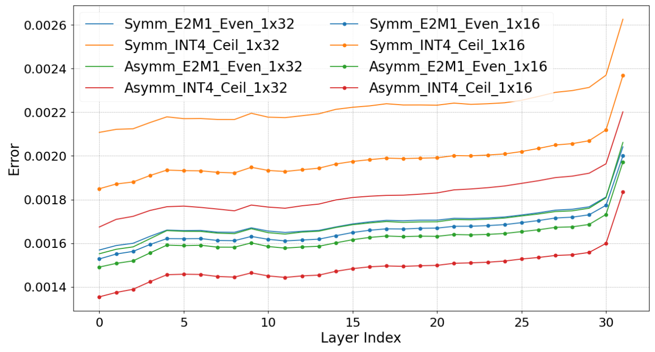
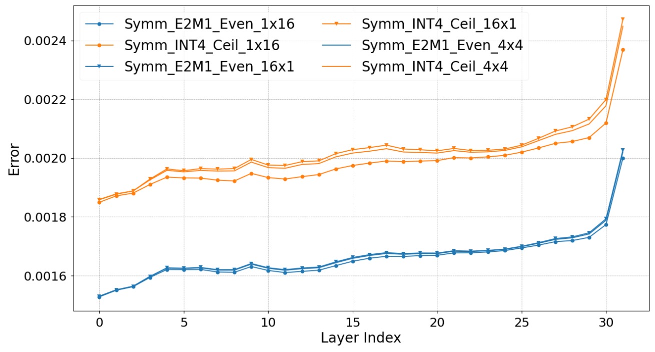


Fig. 9: Impact of block size reduction (from 32 to 16) on the R-MSE of the first weight tensor across all MLP layers for E2M1_Even and INT4_Ceil formats under symmetric and asymmetric scaling.

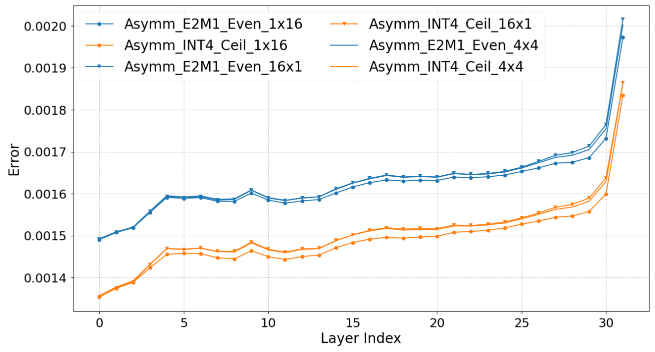
Under symmetric scaling (orange lines), small blocks significantly lower R-MSE, achieving approximately $6.47\times$ the improvement seen with E2M1. With asymmetric scaling (red lines), the reductions in R-MSE are even more pronounced, reaching up to $1.36\times$ better results than symmetric INT4. These trends are consistent across activations and gradients, and the improvements are due to INT4's narrower dynamic range, where smaller block sizes and asymmetric scaling better capture local variations and mitigate outliers.

Insight 5: Symmetric scaling with E2M1 has the least benefit from block size reduction, while both asymmetric scaling and INT4 gain significantly.

Figure 10 and 11 analyze the impact of block organization (row-wise, column-wise, and 2D block-wise) on the R-MSE for weights and activations, respectively, comparing E2M1_Even and INT4_Ceil formats under symmetric (a) and asymmetric (b) scaling. From Figure 10, weight tensors show no benefits from column-wise blocks (16×1) or 2D blocks (4×4), regardless of whether symmetric or asymmetric scaling is used. Similarly, gradients exhibit trends consistent with weights. In contrast, Figure 11 reveals that activation tensors experience substantial R-MSE improvements with column-wise blocks, particularly under asymmetric quantization. This difference arises from the columnar structure of transformer models, where activations for a given token are stored in a column. This layout creates concentrated value distributions, allowing column-wise blocks to align scaling factors more effectively with the data. However, column-wise memory access is inefficient due to non-contiguous layouts, requiring large strides that reduce performance. 2D block-wise scaling mitigates this issue by combining aspects of column-wise scaling with vectorized memory access, while also supporting tensor partitioning in parallel computation to reduce communication overhead. The absence of quantization benefits for weights and gradients suggests that their distributions are less affected by block organization. These trends persist across training steps, highlighting the unique scaling requirements of activations compared to weights and gradients.



(a) Applying symmetric scaling under different block organizations.



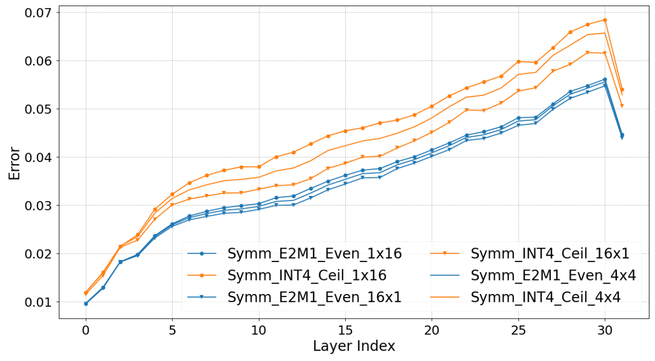
(b) Applying asymmetric scaling under different block organizations.

Fig. 10: Impact of block organization (row-wise vs. column-wise vs. 2D block-wise) on R-MSE for the first weight tensor across all MLP layers, comparing E2M1_Even and INT4_Ceil formats under symmetric and asymmetric scaling.

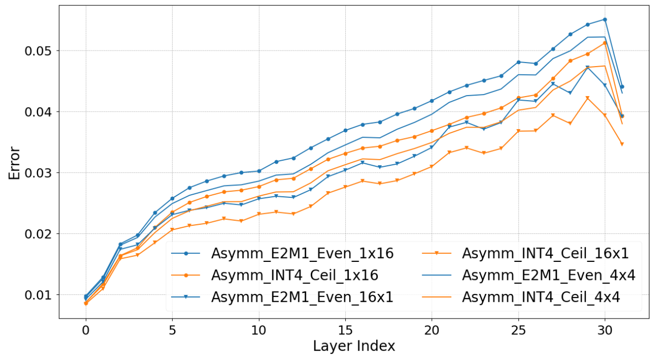
Insight 6: Column-wise blocks enhance activation quantization, while 2D blocks strike a balance between precision and efficiency. However, neither approach yields significant improvements in weight or gradient quantization.

B. *Effectiveness of 4-bit MX Training*

Figure 12 presents the training loss curve of the Llama3 7B model over an extended number of steps, covering 100 million tokens. Two 4-bit MX configurations are evaluated: 1) the first employs E2M1 with Even scaling and uses stochastic rounding for gradient quantization; 2) the second utilizes Ceil scaling, E2M1 on weights and activations, and E3M0 with stochastic rounding on gradients. Both configurations perform well during training, with loss curves closely tracking the FP8 and BF16 baselines for most of the process. While slight divergences appear between 22 and 33 million tokens, the loss curves eventually stabilize and converge, achieving an average loss gap of just 0.02 compared to the baselines by the end of training. These results highlight the promise of 4-bit quantization for pre-training tasks under carefully selected configurations. However, when training is resumed from a pre-trained checkpoint, purely 4-bit configurations struggle to maintain accuracy, as shown in Figure 13. In such scenarios, employing mixed-precision strategies that integrate 4-bit and 6-bit quantization provides a more stable and robust solution,



(a) Applying symmetric scaling under different block organizations.



(b) Applying asymmetric scaling under different block organizations.

Fig. 11: Impact of block organization (row-wise vs. column-wise vs. 2D block-wise) on R-MSE for the second activation tensor across all MLP layers, comparing E2M1_Even and INT4_Ceil formats under symmetric and asymmetric scaling.

effectively striking a balance between numerical accuracy and computational efficiency. This finding implies that tailoring quantization strategies to distinct training phases can help accommodate diverse precision requirements.

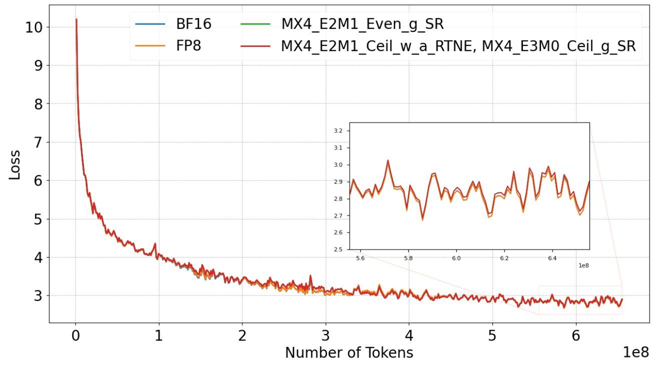


Fig. 12: Training loss of Llama3 7B over extended steps using symmetric quantization with E2M1 and E3M0 formats.

V. Conclusion

In this paper, we presented the first comprehensive study on microscaling (MX) quantization for LLM training, with a focus on the MXFP4 format. We systematically explored the critical design choices and summarized the insights as follows:

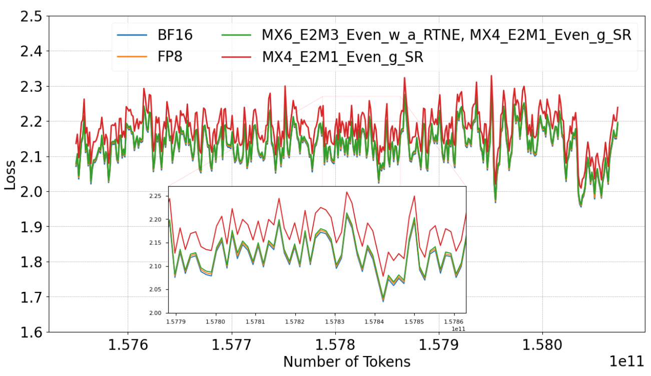


Fig. 13: Training loss of Llama3 7B from a checkpoint trained on 157 billion tokens, comparing purely 4-bit and mixed 4-bit and 6-bit configurations.

- **Data Types and Rounding Modes**: E2M1 works well with both Ceil and Even rounding but achieves better results with Even rounding. E3M0 is effective for gradient quantization only with Ceil rounding. INT4 also requires Ceil rounding to perform well. Stochastic Rounding (SR) is essential for gradient quantization, while RTNE remains effective for weights and activations.
- **Symmetric vs. Asymmetric Scaling**: Asymmetric scaling significantly benefits INT4 quantization by handling skewed data distributions, whereas E2M1 exhibits minor improvements due to its higher dynamic range.
- **Block Granularity and Organization**: Reducing block size improves quantization accuracy, with INT4 under asymmetric scaling showing the largest gains. Column-wise blocks greatly enhance activation quantization by aligning scaling factors with the token dimension in transformer-based model architectures, while 2D blocks provide moderate improvements and better flexibility to transpose operations during training.
- **Training Performance**: Carefully selected 4-bit configurations achieve loss curves closely matching FP8 and BF16 baselines when training from scratch. However, when training from pre-trained checkpoints, mixed-precision of 4-bit and 6-bit quantization better maintains accuracy alignment with FP8 and BF16.

These findings demonstrate the promise of low-precision formats in achieving competitive LLM training accuracy, underscoring the necessity of data-aware and phase-specific quantization configurations. To further optimize low-precision training accuracy, we plan to scale experiments to larger models with extended durations, and further refine design choices during different pre-training or fine-tuning stages for enhanced numerical fidelity and training stability.

Nevertheless, the successful application of low-precision numerical formats requires holistic HW/SW co-design to guarantee the consistent support of numerical features across the system stack, from the tensor/vector operator support in hardware, kernel implementations, to user interfaces in mainstream ML frameworks, etc. For example, asymmetric scaling will require extra hardware support in handling the offsets. Finer-granularity and 2D scaling methods also need extra hardware and kernel support for both GEMM and conversion operators. This paper discusses the performance potential from low-precision LLM training, as well as the critical quantization recipes to guarantee the numerical accuracy. We expect our investigations to provide valuable guidelines on MX support in both hardware and software to deliver the best performance and accuracy trade-offs for LLM training systems.

References

[1] B. Chmiel, R. Banner, E. Hoffer, H. Ben-Yaacov, and D. Soudry, "Accurate neural training with 4-bit matrix multiplications at standard formats," in *The Eleventh International Conference on Learning Representations*, 2023.

[2] B. Chmiel, L. Ben-Uri, M. Shkolnik, E. Hoffer, R. Banner, and D. Soudry, "Neural gradients are near-lognormal: improved quantized and sparse training," in *International Conference on Learning Representations*, 2021.

[3] M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis, "Stochastic rounding: implementation, error analysis and applications," *Royal Society Open Science*, vol. 9, no. 3, p. 211631, 2022.

[4] B. Darvish Rouhani, R. Zhao, V. Elango, R. Shafipour, M. Hall, M. Mesmakhosroshahi, A. More *et al.*, "With shared microexponents, a little shifting goes a long way," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.

[5] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[6] M. Fishman, B. Chmiel, R. Banner, and D. Soudry, "Scaling fp8 training to trillion-token llms," 2025. [Online]. Available: https://arxiv.org/abs/2409.12517

[7] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen *et al.*, "A study of bfloat16 for deep learning training," 2019. [Online]. Available: https://arxiv.org/abs/1905.12322

[8] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," in *International Conference on Learning Representations*, 2018.

[9] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, N. Mellempudi, S. Oberman, M. Shoeybi, M. Siu, and H. Wu, "Fp8 formats for deep learning," 2022. [Online]. Available: https://arxiv.org/abs/2209.05433

[10] NVIDIA, "Nvidia blackwell architecture technical brief," 2024, accessed: 2025-02-23. [Online]. Available: https://resources.nvidia.com/en-us-blackwell-architecture

[11] Open Compute Project, "Ocp microscaling formats (mx) v1.0 specification," 2023. [Online]. Available: https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf

[12] B. D. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, M. Cornea, E. Dellinger, K. Denolf *et al.*, "Microscaling data formats for deep learning," 2023. [Online]. Available: https://arxiv.org/abs/2310.10537

[13] torchao maintainers and contributors, "torchao: Pytorch native quantization and sparsity for training and inference," Oct. 2024. [Online]. Available: https://github.com/pytorch/torchao

[14] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[15] S. Wiedemann, T. Mehari, K. Kepp, and W. Samek, "Dithered backprop: A sparse and quantized backpropagation algorithm for more efficient deep neural network training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 720–721.

[16] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li, "Pytorch fsdp: Experiences on scaling fully sharded data parallel," 2023. [Online]. Available: https://arxiv.org/abs/2304.11277