

MX-SAFE: Versatile Inference- and Training-Proof Microscaling Format with On-the-Fly Exponent and Mantissa Bit Allocation

Dahoon Park^{*†}, Jahyun Koo^{*‡}, Sangwoo Hwang[†], and Jaeha Kung^{§†}

[†] School of Electrical Engineering, Korea University, South Korea

[‡] Department of Electrical Engineering and Computer Science, DGIST, South Korea

Abstract—As the demand for deep learning grows, cost reduction through quantization has become essential for both training and inference. In 2022, the Open Compute Project (OCP) consortium standardized narrow precision formats for deep learning, called the microscaling (MX) format. The MX format is a hardware-friendly dynamic quantization scheme that effectively reduces the data size by sharing an 8-bit exponent across multiple operands. The MX format can be categorized into two types with their own strengths: (i) MXINT which focuses on a high precision consisting only of mantissa bits and (ii) MXFP which focuses on a wider dynamic range by allowing local exponent bits. In this work, we present a versatile MXFP format, called MX-SAFE (MXSF in short), that adaptively uses two modes, i.e., a wider mantissa mode (FP8_E2M5) and a subnormal FP mode (FP5_E3M2), to support both training and direct-cast inference. Furthermore, we propose a tile-based block design to increase hardware efficiency by reducing the burden of re-quantization process during the training with the MXSF format. Owing to the use of the proposed MXSF format, 0.05%/11.1% and 3.55%/3.57% improvements in accuracy, on average, for inference/full-training compared to MXFP8_E2M5 and MXFP8_E4M3 are observed, respectively. Moreover, we present a training-inference accelerator that supports the MXSF format and it achieves similar accuracy to the BF16 baseline while using 24.9% less total energy consumption.

Index Terms—Direct-cast inference, hardware accelerator, hybrid microscaling format, low-precision training

I. INTRODUCTION

As the size of deep neural networks continues to grow, quantization has become one of the most effective compression techniques. The quantization technique can be broadly categorized into static and dynamic quantization. The static quantization has the advantage of being able to compress inputs and weights based on pre-computed statistics, enabling efficient computation and reduced memory requirement. However, as recent models have become more complex, some intermediate operations require dequantization process to floating point representation (e.g., softmax and layer norm), and each model has

This work was partially supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korea government (MSIT) under Grant RS-2023-00229849 and Grant RS-2025-25442405; in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant NRF-2023R1A2C2006290. The EDA tool was supported by the IC Design Education Center (IDEC) in South Korea.

* D. Park and J. Koo are equally contributed authors.

§ J. Kung is the corresponding author (email: jhkung@korea.ac.kr).

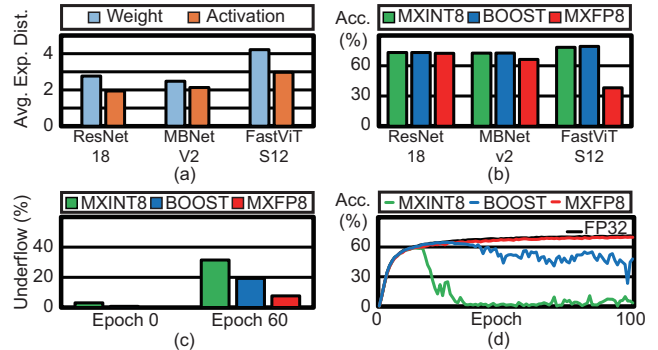


Fig. 1. Impact of various MX formats on inference (a-b) and training (c-d): (a) average distance to the max exponent within a block during inference, (b) direct-cast inference accuracy on ImageNet-1K, (c) ratio of weight gradient underflows at FastViT-T8's *stages.0.blocks.0.mlp.fc1* layer during training, and (d) training curve for FastViT-T8.

its own optimal statistical data. Particularly, weight parameters keep changing during training which makes it very challenging to use the static quantization. The dynamic quantization, on the other hand, extracts tensor statistics on the fly by inspecting the incoming data and then performs quantization. This allows for more accurate quantization [1], [2], but it comes at the cost of running quantization-dequantization steps.

Recently, microscaling (MX) format has been standardized as a new data format by Open Compute Project [3] which simplifies the dynamic quantization process, efficiently converting FP32 computations to low-bit computations. The MX format is currently supported by commodity deep learning hardware such as NVIDIA Blackwell Tensor Cores [4], Microsoft MAIA100 [5], and Tenstorrent Wormhole [6], as well as custom accelerators [7]–[11]. With various MX formats being proposed, however, it is difficult for us to select the best MX format which varies depending on the task, either inference or training. For direct-cast inference¹, the distance between the max exponent and each element's exponent within a block is small (Fig. 1(a)), making the mantissa bit-width critical. Consequently, MXINT8 and BOOST (MXFP8_E2M5) [9] outperform MXFP8_E4M3 (Fig. 1(b)). However, as shown in Fig. 1(c), a significant portion of small gradient values become zeros (i.e., underflows) for MX formats with limited local

¹Directly casting FP32 elements to lower bit-width representation, e.g., MXINT8 or MXFP8, without any training and calibration.

exponent bits. Thus, only MXFP8_E4M3 format which can represent small values using local exponent bits, on top of the shared exponent, shows the similar training curve to FP32 while other MX formats fail to converge (Fig. 1(d)).

These motivational experiments on both inference and training imply that each MX format type has its own benefits (MXINT and BOOST for a higher precision and MXFP for a wider dynamic range), but no universal solution exists. Therefore, for the first time, we present a versatile 8-bit MXFP format, named MXSF, that guarantees safe direct-cast inference and full training. Our MXSF is defined based on the quantitative error analysis, so that we can switch between MXFP formats with wide mantissas and wide exponents on the fly. To test the validity of the proposed MXSF, a wide range of deep learning models, from CNNs to transformers, and both inference and training tasks are evaluated.

Our main contributions can be summarized as follows:

- **Quantitative analysis on MXINT/MXFP:** We quantify the quantization error that occurs during conversion from FP32/BF16 to MXINT and MXFP. Our analysis shows that the optimal MX format changes with respect to the exponent distance to the shared exponent within a block.
- **Safe MX format:** We propose a novel MX-SAFE format, i.e., MXSF in short, which safely supports direct-cast inference and full-training/fine-tuning. The MXSF format dynamically stores an element with MXFP8_E2M5 or with MXFP5_E3M2 that targets to reduce both quantization errors and underflows.
- **MX-SAFE hardware accelerator:** We design a MAC unit that handles MXSF formats and implement a systolic tensor array built out of the proposed MXSF-aware MACs.

II. BACKGROUNDS

A. Microscaling (MX) Data Format

Recently, microscaling (MX) data formats [3] have been presented showing their effectiveness in direct-cast inference or training of various deep learning models. The original MX format packs 32 data into a block (i.e., $\text{block_size} = 32$), shares one exponent per block (S_e), and aligns each element in the block according to the shared exponent. While sharing the global exponent, each element in the block can be represented by either an integer format, e.g., MXINT8, or a floating point format, e.g., MXFP8_E4M3 or MXFP8_E5M2². One may also choose to use a lower-bit floating point representation, e.g., MXFP4_E2M1 or MXFP6_E2M3, MXFP6_E3M2.

B. Optimal MX Format for Inference & Training

While the MX format has many different variants, there is an ambiguity in that the optimal type/format depends on the task at hand. During inference, tensor distributions show relatively smaller variances compared to gradients in training. In other words, local exponents within a block are close to each other making MXINT8 and MXFP8_E2M5 represent the

tensors more accurately compared to MXFP8_E4M3 owing to wider mantissa bits. On the other hand, during training, gradients have high variance, leading to larger differences between local exponent values within a block, which causes many underflows if exponent bits are not properly assigned. Thus, the model trained in MXFP8 with wider exponent bits outperforms the one trained in MXINT8. However, supporting both INT and FP types in hardware introduces high overheads. While there are some prior works demonstrating training with MXFP4_E2M1 [12], [13], they have notable drawbacks. First, [12]-based training results in suboptimal training performance in CNNs (Section III-B). Furthermore, the method presented in [13] requires additional hardware blocks for tensor rotation which incurs latency overhead [14]. Also, both works perform BF16 computations on $Q \cdot K^T$ and $\text{Attn} \cdot V$ in attention layers. Thus, in this work, we *stick to an 8-bit MX format for all computations and develop a new MX format* that appropriately combines advantages of both MX data types with minimal hardware overhead.

III. QUANTITATIVE ANALYSIS ON MX FORMATS

A. Analytical Comparison Between MXINT and MXFP

Prior to proposing a new MX data format, we start by analytically comparing MXINT and MXFP formats. The following equation provides the quantized value in MXINT for a given floating point value x .

$$MXINT(x) = s_x \cdot 2^{S_e} \cdot \left\lfloor \frac{1 \cdot m_x \cdot 2^{(m_i-2)}}{2^{(S_e-e_x)}} \right\rfloor \cdot 2^{-(m_i-2)}, \quad (1)$$

where $x = s_x \cdot 2^{e_x} \cdot 1 \cdot m_x$ in FP32/BF16, s_x is the sign, e_x is the exponent, and m_x is the mantissa value of x . The S_e is the shared exponent within the block containing x and m_i is the MXINT's mantissa bit-width. On the other hand, the same value x can be represented by the MXFP format as follows:

$$MXFP(x) = \begin{cases} FP_{norm}(x) & \text{if } x_{le} > 0, \\ FP_{subnorm}(x) & \text{if } x_{le} \leq 0, \end{cases} \quad (2)$$

where the local exponent $x_{le} = E - (S_e - e_x)$. Here, E is the maximum local exponent value, i.e., $2^{e_f} - 1$, where e_f is the MXFP's local exponent bits. The FP_{norm} and $FP_{subnorm}$ are defined as follows:

$$FP_{norm}(x) = s_x \cdot 2^{S_e-E} \cdot 2^{x_{le}} \cdot \left\lfloor 1 \cdot m_x \cdot 2^{m_f} \right\rfloor \cdot 2^{-m_f}, \quad (3)$$

$$FP_{subnorm}(x) = s_x \cdot 2^{S_e-E} \cdot \left\lfloor 0 \cdot m_x \cdot 2^{m_f+x_{le}} \right\rfloor \cdot 2^{-m_f}, \quad (4)$$

where m_f is the MXFP's mantissa bits.

Based on Eq. (1) and Eq. (3-4), the maximum quantization errors of MXINT and MXFP formats can be formulated by

$$\Delta_{MXINT} = 2^{S_e-(m_i-2)} \cdot 2^{(S_e-e_x)-(m_i-2)}, \quad (5)$$

$$\Delta_{MXFP} = 2^{e_x-m_f} \cdot 2^{-\min(x_{le},0)-m_f}. \quad (6)$$

For both cases, the error occurs from the round function which is the second multiplicative term in Eq. (5-6). In case of MXINT, error occurs due to the difference between the

²Here, 'E' and 'M' denote (local) exponent bits and mantissa bits, respectively. The total bit-width N becomes $E + M + 1$ where 1-bit is for sign.

TABLE I
MEAN SQUARED ERRORS OF DIRECTLY CASTING
INTO DIFFERENT MX FORMATS IN VARIOUS MODELS

Model	ResNet-18		MobileNetV2		FastViT-T8	
	Act. (10^{-4})	Weight (10^{-6})	Act. (10^{-4})	Weight (10^{-6})	Act. (10^{-4})	Weight (10^{-4})
MXINT8	1.31	6.13	11.2	3.41	1.27	1.48
MXFP8_E2M5	1.14	3.41	8.35	2.29	0.79	1.00
MXFP8_E4M3	17.3	42.5	120	31.0	10.3	13.1
MXSF (Proposed)	1.40	5.07	10.7	3.06	1.10	1.17

* Best and second best are marked in red and blue, respectively.

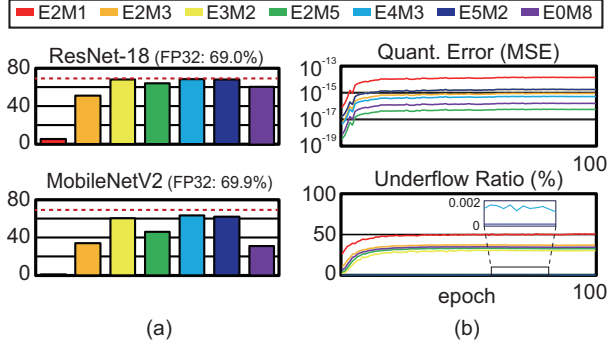


Fig. 2. (a) Training performance with various microscaling formats. The red dotted line represents FP32 training accuracy. (b) Quantization error and underflow ratio during training (target: ResNet-18’s *layer2.1.conv2.grad*)

shared exponent S_e and the exponent of x (e_x), as well as the supported mantissa bits (m_i). In contrast, in MXFP, it has the local exponent x_{le} , and as long as x_{le} does not fall below zero, the rounding error is only caused by the supported mantissa bits (m_f). Based on Eq. (5-6), we can compare MXINT8 and MXFP8_E2M5 in terms of quantization error at various ‘ $S_e - e_x$ ’ distances. Only when $S_e - e_x = 0$, MXINT8 provides a smaller error than MXFP8_E2M5. If $S_e - e_x = 1$, both experiences the same quantization error. When $S_e - e_x$ exceeds 1, then MXFP8_E2M5 introduces a smaller quantization error. Since both weight and activation tensors exhibit the average exponent distance ($S_e - e_x$) greater than 2 (Fig. 1(a)), MXFP8_E2M5 is expected to have a lower quantization error than MXINT8, as summarized in Table I. Therefore, we establish MXFP8_E2M5 as the baseline for inference. The following section will discuss how this MXFP8 format can be modified to make it suitable for training.

B. Analysis on Required Bit-precision in Training

According to Table I, MXINT8 and MXFP8_E2M5 have lower quantization error than MXFP8_E4M3. However, those two MX formats paradoxically exhibit training instability as shown in Fig. 1(d). This is mainly because the occurrence of underflows has a more significant impact on training stability than quantization errors. Fig. 2(a) illustrates the results of various MX-format training runs³ on ResNet-18 [15] and MobileNetV2 [16]. For both benchmarks, at least three exponent bits were required for stable training. However, for those MX formats with wider exponent bits, the quantization error increases due to narrower mantissa bits, which results in a noticeable accuracy degradation in MobileNetV2.

³Here, MXFP4 is based on TetraJet [12] Q-EMA algorithm.

Algorithm 1: MXSF Conversion

Data: X : floating point numbers within a block
Result: \hat{X} : converted MXSF data for tensor X

- 1: $S_e \leftarrow \text{floor}(\log_2(\max(|X|)));$
- 2: $X_e \leftarrow \text{floor}(\log_2(|X|));$
- 3: **if** ($S_e - X_e < 3$) **then**
- 4: $\hat{X} \leftarrow \text{MXFP}(X, e_f = 2, m_f = 5, \text{bias} = 3);$
- 5: **else**
- 6: $\hat{X} \leftarrow \text{MXFP}(X, e_f = 3, m_f = 2, \text{bias} = 10);$
- 7: **end**

To investigate the cause of training instability in detail, we analyzed the quantization error and underflow ratio of activation gradients at the *layer2.1.conv2* layer in ResNet-18. The analysis was performed by converting the gradients into various MX formats and monitoring the metrics across training epochs. The results shown in Fig. 2(b) reveal that while MXINT8 (E0M8) and BOOST (E2M5) show lower quantization error than MXFP8_E4M3, they have significantly higher underflow ratios, which prevents the model from stable training. This suggests that even gradients with very small magnitudes are critical to training performance and must be preserved. Based on these analyses, we propose MXSF, a unified MX format that deals with both underflows and quantization errors by dynamically adjusting exponent and mantissa bits based on the criteria explained in Section III-A.

IV. MX-SAFE: VERSATILE MICROSCALING FORMAT

A. Proposed MXSF Data Format

The proposed MXSF format, named after a safe MX format, is designed to support two FP formats in a single block. The main goal of supporting two data types in a single block is to minimize quantization errors and reduce underflows. This makes the proposed MXSF versatile, so that we can use this format in any AI deployment stages. In this work, we focus on designing the MXSF format that supports a narrow exponent + large mantissa (E2M5), which shows similar accuracy to FP32 for inference, and a wider exponent with extra bias + small mantissa (E3M2 + bias), which performs well during training by suppressing underflows.

To accommodate both within a single block, we propose a format that allocates more exponent bits, on the fly, for elements with a large exponent gap (i.e., $S_e - e_x$) as illustrated in Fig. 3. Specifically, we repurpose the subnormal representation in MXFP8_E2M5, where local exp $x_{le} = 00$ (the 2nd and the last elements in Fig. 3), to better represent small numbers, denoted E3M2 (or sub-FP), using the remaining 5 bits. In the original MXFP8_E2M5 format, values with an exponent gap of 3 or larger ($S_e - e_x \geq 3$) fall into the subnormal category. In our MXSF, this region is represented by using the sub-FP format, effectively extending the minimum exponent value from -3 down to -9. The addition of E3M2 format extends the representable value range without overlapping the range of E2M5 with proper biasing (bias = 10). This allows our

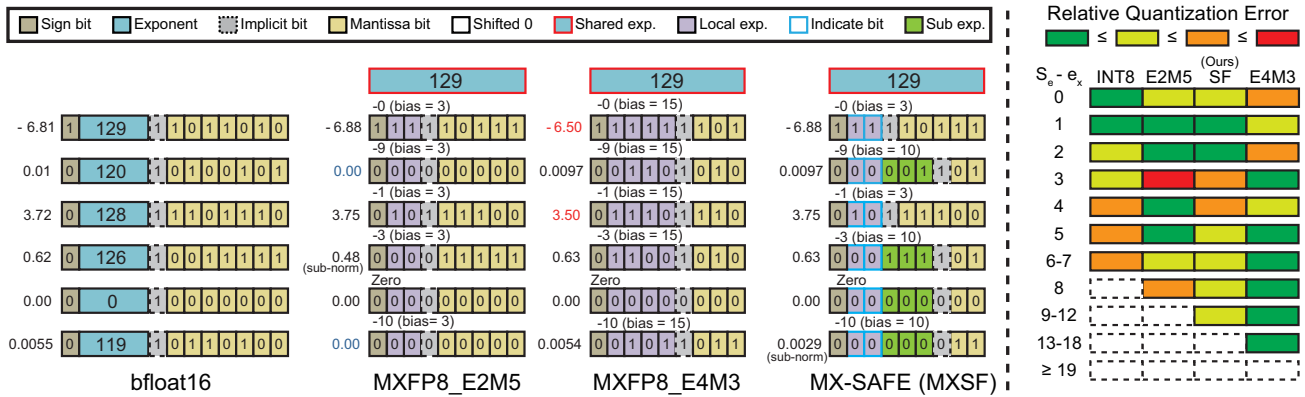


Fig. 3. (Left) Bfloat16 as a baseline and various microscaling data formats, i.e., MXFP8_E2M5, MXFP8_E4M3, and proposed MX-SAFE (MXSF). The blue values in MXFP8_E2M5 represents the underflow and the red values in MXFP8_E4M3 show conversion errors larger than 0.2. (Right) Visualization of relative quantization error with respect to a distance between the shared exponent and the local exponent ($S_e - e_x$). The blank dashed box denotes underflow.

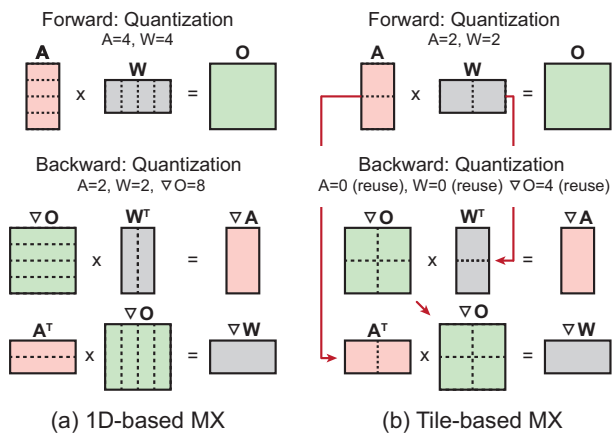


Fig. 4. The forward and backward passes for a linear layer during training. (a) 1D-based MX block (1×4) and (b) 2D tile-based MX block (2×2). The red arrows highlight the reuse of quantized MX blocks.

modified representation to cover a dynamic range similar to that of MXFP8_E3M4, slightly lower than E4M3. As shown in Fig. 3, when the exponent gap is 2 or less, MXSF behaves same as MXFP8_E2M5, preserving direct-cast inference accuracy. When an element within a block has an exponent gap greater than 2, local exponent becomes ‘00’ indicating the use of E3M2 dynamically extending the dynamic range for small values, improving the training stability. The conversion process for MXSF is described in Alg. 1.

B. MX Block Tiling for Inference/Training

As shown in Fig. 4(a), the commonly used 1D block design in the current MX format is suited at the forward-only computation flow of inference. In this setting, quantized weights remain unchanged and activations are not reused, making the 1D block structure suitable without any issues. However, during the training process, particularly in the backward pass, both weights and activations are transposed and used for computing gradients. Due to the transpose operation, weight/activation tensors need to be dequantized and re-quantization is needed along the transposed dimension, which

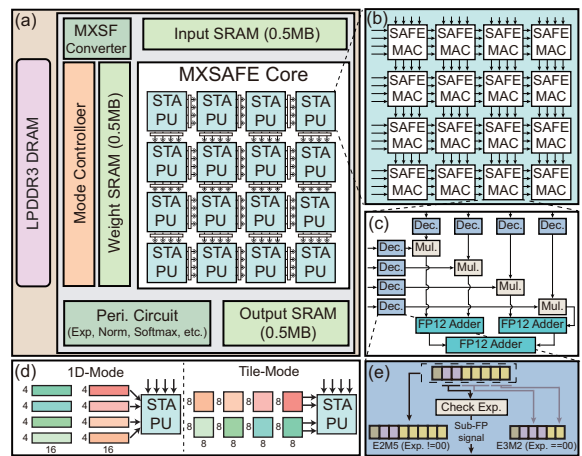


Fig. 5. Overview of MX-SAFE accelerator architecture.

incurs latency overhead. Fig. 4 shows a simple example with the number of MX quantization process for each tensor.

To address this limitation, as illustrated in Fig. 4(b), we propose 2D tile-based MX block design. This 2D tiling strategy makes us to maintain the quantized tensor in its original shape and reuse it during the backward pass without additional dequantize/quantize step. Accordingly, we design a training-inference accelerator using our versatile MXSF that supports both 1D and 2D MX blocks, maximizing overall hardware efficiency. The 1D block tiling is also supported in the hardware for the inference with batch size of 1.

V. MXSF-BASED MULTI-FORMAT SYSTOLIC TENSOR ARRAY ACCELERATOR

A. MX-SAFE Accelerator

Fig. 5(a) shows overall architecture of the MX-SAFE accelerator, which is based on a systolic tensor array (STA [17]). The STA generalizes the conventional systolic array by grouping multiple MACs to form a computing block, where each MAC contains multiple multipliers (*vectorized*). Grouping multiple MACs as a block, i.e., processing unit (PU), reduces the number of registers inserted between neighboring MACs. MX-SAFE core contains 16 PUs in a 4×4 configuration,

where these PUs are pipelined for the systolic execution. Then, each PU contains 4×4 MAC units which perform sub-matrix multiplication in parallel (Fig. 5(b)). Since each MAC unit has four multipliers, 16 inputs and 16 weights are provided at each cycle for parallel execution (Fig. 5(c)). As illustrated in Fig. 5(d), we support taking both 1D and 2D MX blocks as input operands. The 1D mapping assigns four different blocks to a single STA PU, whereas tiled mapping assigns two blocks. The tiled approach creates a minor accumulation overhead in the STA but advantageously allows for a dataflow design that does not require reshaping 2D tiles into a 1D format.

B. MXSF-aware MAC Unit

Fig. 5(c) shows SAFE-MAC, a new MAC unit designed to support the MXSF format. Unlike conventional formats, MXSF represents two distinct numerical formats, E2M5 and E3M2, within a single block. Consequently, each element must be decoded to its corresponding value prior to computation. The decoding logic (Fig. 5(e)) interprets the data as the E3M2 format if its 2nd and 3rd MSBs are both zero; otherwise, it is treated as the E2M5 format. The SAFE-MAC unit processes four input-weight pairs per cycle to compute one partial sum. To compute on any of two decoded MXSF formats, each data pair is first multiplied by using an E4M5-based multiplier (Mul.) that fully covers E2M5 and E3M2. The four multiplier outputs are then accumulated to calculate the partial sum using an FP12_E4M7 adder tree. We select FP12 adders, which provide enough bit coverage, instead of BF16 adders to minimize the hardware overhead coming from the adder tree in the SAFE-MAC.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

1) *Software Setup*: We modified MX implementation from Microsoft⁴ to evaluate the effectiveness of our MXSF format on direct-cast inference and training. The implementation includes MXINT8, MXFP8_E4M3, block minifloat from BOOST (i.e., MXFP8_E2M5 [9]) and our MXSF. The BOOST uses a 48×48 block, but for a fair comparison, we fixed all MX format’s block sizes to 1×64 for inference and 8×8 for training (favorable choice for BOOST). Utilizing the PyTorch framework, we copied pre-trained models from “timm” for a vision task and “Hugging Face [18]” for LLM and VLM tasks in our experiments.

2) *Direct-cast Inference*: To compare accuracy of various MX formats on direct-cast inference (FP32→MX), we select an image classification task and language generation tasks. For the image classification, we tested ResNet-18 [19], MobileNetV2 [16] and MobileNetV4-based MNv4-Conv-S [20] as CNN benchmarks, while DeiT-B [21] and Swin-S [22] are used as Vision Transformer (ViT) benchmarks. Also, EfficientViT-B3 [23] and FastViT-S12 [24] are selected as CNN+ViT benchmarks. As a dataset, a widely used ImageNet-1K validation dataset is selected. For the language generation

TABLE II
DIRECT-CAST INFERENCE VISION & LLM ZERO-SHOT TASK
(MXFP8: MXFP8_E4M3, BOOST: MXFP8_E2M5)

Vision Task (dataset: ImageNet-1K)					
Model	Baseline	MXINT8	MXFP8	BOOST	MXSF
CNN architecture					
ResNet-18 (Acc.) ↑	73.14	73.15	72.27	73.14	73.05
MobileNetV2 (Acc.) ↑	72.95	72.48	66.27	72.51	72.46
MNv4-Conv-S (Acc.) ↑	74.58	73.18	61.80	73.89	73.60
Vision transformer (ViT) architecture					
DeiT-B (Acc.) ↑	81.97	81.91	81.75	81.95	81.94
Swin-T (Acc.) ↑	81.36	81.35	81.09	81.32	81.36
Hybrid ViT architecture					
EfficientViT-B3 (Acc.) ↑	83.35	82.69	74.64	83.04	82.79
FastViT-S12 (Acc.) ↑	79.89	77.91	38.25	79.00	78.84
LLM Task (Model: Llama3.2-3B)					
Task	Baseline	MXINT8	MXFP8	BOOST	MXSF
WikiText2 (PPL) ↓	9.27	9.30	9.36	9.28	9.28
BoolQ (Acc.) ↑	72.94	72.42	70.46	72.78	73.03
ARC-e (Acc.) ↑	74.49	74.71	74.07	74.33	74.28
ARC-c (Acc.) ↑	42.66	42.41	41.98	42.06	42.32

* Best and second best are marked in red and blue, respectively.

task, we used EleutherAI’s lm-evaluation-harness [25] to conduct experiments on the LLaMA-3.2-3B [26]. We evaluate various data formats on three question-answering tasks, i.e., Boolean Question (BoolQ) and AI2 Reasoning Challenge (ARC-Easy and ARC-Challenge), as well as one language modeling task, WikiText2.

3) *Model Training*: We report the training quality of various MX formats on two different tasks: a vision task and a multimodal task. To compare the full-training performance, we selected three different models, i.e., ResNet-18, MobileNetV2, and DeiT-Tiny for ImageNet-1K classification. We used eight RTX 3090 GPUs with a batch size of 128 per GPU, trained for 100 epochs, and selected the AdamW optimizer with a learning rate of $1e-3$ and a cosine decay scheduler. To compare the training performance on a visual language model (VLM) using the MX format, we trained nanoVLM-222M [27] from Hugging Face using an open-source dataset. We set batch_size = 256 and conducted training on an RTX6000 Blackwell GPU based on the detailed configuration from the original nanoVLM GitHub repository⁵.

4) *Hardware Design*: All hardware modules in Fig. 5 for MXSF are implemented in RTL and synthesized with 65nm CMOS technology running at 250MHz. For performance comparison, we implemented two baselines. The first baseline is a BF16 baseline accelerator, which has the same STA configuration. The second baseline uses a more efficient MXFP4 format for the main STA core. However, a small 8×8 BF16 systolic array is included, as certain operations, such as $Q \cdot K^T$, require BF16 to maintain accuracy. The area and power consumption for these accelerators are estimated using Synopsys Design Compiler.

B. Accuracy on Direct-cast Inference

Table II summarizes the direct-cast inference accuracy on the ImageNet classification and LLM zero-shot inference tasks. In the domain of vision tasks, the MXFP8_E4M3 format exhibits a notable performance degradation relative to other

⁴<https://github.com/microsoft/microscaling>

⁵<https://github.com/huggingface/nanoVLM/tree/v0.1>

TABLE III
ACCURACY ON TRAINING VISION TASKS (DATASET: IMAGENET-1K)

Model	Baseline	MXINT8	MXFP8	BOOST	MXSF
ResNet-18	69.0	56.9	65.3	62.2	68.4
MobileNetV2	69.9	31.1	63.2	46.2	69.5
DeiT-Tiny	67.2	50.3	65.4	63.0	66.7

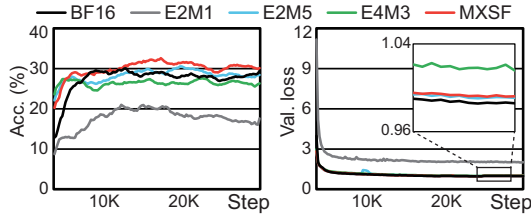


Fig. 6. Training curve for nanoVLM-222M using various MX formats: accuracy (left) and validation loss (right) on the MMStar [28] benchmark.

8-bit MX formats. This degradation is primarily attributed to its limited mantissa bit-width, which induces substantial quantization errors. The effect is particularly pronounced in lightweight models, where the resulting accuracy degradation can exceed 8%. Conversely, the MXINT8, BOOST, and MXSF formats demonstrate superior robustness by effectively mitigating performance loss, constraining accuracy degradation to within 1.1%. Moreover, in models where values satisfying $S_e - e_x \geq 3$ are prevalent, i.e., hybrid ViTs, both BOOST and MXSF have been observed to outperform MXINT8, yielding higher final accuracy. For LLM tasks, other data formats still show slightly better performance compared to MXFP8_E4M3. Based on these experiments, we show that the MXSF achieves similar accuracy as BF16 for the direct-cast inference.

C. Accuracy on Model Training

Other than inference, we also tested the stability of the MXSF format on training tasks. Table III summarizes the ImageNet-1K training accuracy of full-training scenarios. Since small gradients are important throughout the entire training process, both the MXFP8_E4M3 and MXSF formats show excellent performance in preventing underflow. Furthermore, by allocating more mantissa bits to values near the shared exponent, the MXSF format improves its resilience to quantization error, resulting in an average of 3.6% higher training accuracy than MXFP8_E4M3 in classification tasks.

As shown in Fig. 6, the nanoVLM training task involves fine-tuning a combination of a pre-trained vision encoder (SigLIP-B [29]) and a small language model (SmolLM2-135M [30]) for the multimodal objective. Unlike the full-training case, fine-tuning is performed on pre-trained weights making significantly small gradients to appear less, which explains why FP8_E2M5 achieves high training accuracy. Nevertheless, even in this scenario, MXSF is observed to deliver slightly better accuracy compared to other MX formats.

D. Hardware Analysis

Based on the area and power consumption in Table IV, we performed a detailed hardware analysis using DeiT-Tiny model on ImageNet. Table IV shows that the MX-SAFE core

TABLE IV
AREA AND POWER BREAKDOWNS OF MX-SAFE ACCELERATOR

Ours (65nm, 250MHz)	Area (μm^2)	Power (mW)
MX-SAFE Core	1,784,575.55 (87.68%)	308.845 (84.18%)
MXSF Converter	14,925.96 (0.75%)	7.09 (1.93%)
Act. Function Unit	10,969.92 (0.55%)	2.61 (0.71%)
Softmax	181,606.47 (9.11%)	24.12 (6.57%)
Norm. Unit	38,267.18 (1.92%)	24.21 (6.60%)

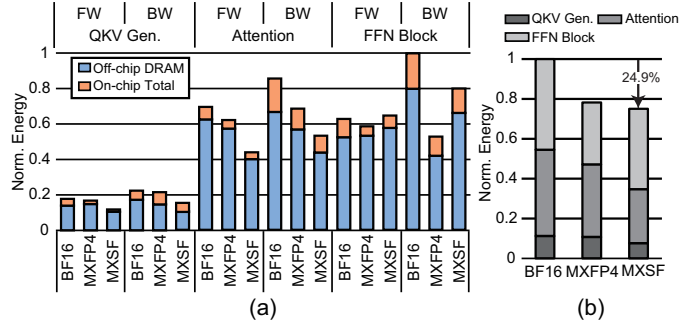


Fig. 7. Energy consumption of DeiT-Tiny during a single-batch training. (a) Breakdowns per encoder block and (b) overall system energy consumption.

is the dominant hardware component, consuming the majority of both the total area (87.68%) and power (84.18%) of the MX-SAFE accelerator. As shown in Fig. 7, the total energy consumption is primarily driven by the off-chip memory access, which accounts for 83.93%. The remaining energy is consumed by the on-chip memory access and the core, contributing 14.92% and 1.15%, respectively. For the energy estimation, we utilized formulas from BitMoD [31] to model the off-chip and on-chip memory accesses. Note that MXFP4 consumes more energy on Attention layers since it uses BF16 for $Q \cdot K^T$ and $Attn \cdot V$. In return, the proposed MXSF format demonstrates a 24.92% reduction in total energy consumption compared to the BF16 baseline. Furthermore, a comparison with the MXFP4 hardware, which includes a small BF16 systolic array, shows that our design is 4.07% more energy-efficient, underscoring the accuracy and the energy efficiency of our MXSF format.

VII. CONCLUSION

In this paper, we proposed MX-SAFE (MXSF) format, which safely supports both inference and training by allowing a large mantissa mode (E2M5) and a high dynamic range mode (E3M2) within the same block. MXSF proves to be an outstanding substitute for BF16 in both training and inference. It results in a negligible accuracy drop compared to BF16—only 0.17% on average for direct cast inference and under 0.5% for training—distinguishing it from other MX formats. Furthermore, to optimize training efficiency, we minimized both the memory usage during training and the quantization process by employing tile-based blocks instead of the 1D blocks used in the existing MX format. Overall, we present MX-SAFE accelerator, i.e., a training-inference accelerator based on the MXSF format. Our MXSF-based hardware accelerator achieves accuracy similar to BF16 baseline while reducing total energy consumption, including off-chip memory accesses, by 24.9% in the DeiT-Tiny training task.

REFERENCES

- [1] Z. Liu *et al.*, “SpinQuant: LLM Quantization with Learned Rotations,” *arXiv:2405.16406*, 2024.
- [2] J. So, J. Lee, D. Ahn, H. Kim, and E. Park, “Temporal Dynamic Quantization for Diffusion Models,” in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.
- [3] B. D. Rouhani *et al.*, “Microscaling Data Formats for Deep Learning,” *arXiv:2310.10537*, 2023.
- [4] A. Tirumala and R. Wong, “NVIDIA Blackwell Platform: Advancing Generative AI and Accelerated Computing,” in *Proceedings of the IEEE Hot Chips Symposium (HCS)*. IEEE Computer Society, 2024, pp. 1–33.
- [5] S. Xu and C. Ramakrishnan, “Inside Maia 100,” in *Proceedings of the IEEE Hot Chips Symposium (HCS)*. IEEE Computer Society, 2024, pp. 1–17.
- [6] J. Vasiljevic *et al.*, “Compute substrate for software 2.0,” *IEEE Micro*, vol. 41, no. 2, pp. 50–55, 2021.
- [7] J. Koo, D. Park, S. Jung, and J. Kung, “OPAL: Outlier-Preserved Microscaling Quantization Accelerator for Generative Large Language Models,” in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, 2024.
- [8] Y.-C. Lo, T.-K. Lee, and R.-S. Liu, “Block and Subword-Scaling Floating-Point (BSFP): An Efficient Non-Uniform Quantization for Low Precision Inference,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.
- [9] C. Guo *et al.*, “BOOST: Block Minifloat-Based On-Device CNN Training Accelerator with Transfer Learning,” in *Proceedings of The IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.
- [10] S. Lee, J. Choi, S. Noh, J. Koo, and J. Kung, “DBPS: Dynamic Block Size and Precision Scaling for Efficient DNN Training Supported by RISC-V ISA Extensions,” in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.
- [11] S. Q. Zhang, B. McDanel, and H. Kung, “FAST: DNN Training Under Variable Precision Block Floating Point with Stochastic Rounding,” in *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 846–860.
- [12] Y. Chen, H. Xi, J. Zhu, and J. Chen, “Oscillation-Reduced MXFP4 Training for Vision Transformers,” in *Proceeding of the International Conference on Machine Learning (ICML)*, 2025. [Online]. Available: <https://openreview.net/forum?id=LUFPNGiCUw>
- [13] A. Tseng, T. Yu, and Y. Park, “Training LLMs with MXFP4,” in *Proceedings of The International Conference on Artificial Intelligence and Statistics (AISTAT)*, 2025. [Online]. Available: <https://openreview.net/forum?id=a8z5Q0WSPL>
- [14] S. Kim, Y. Choi, J. Oh, B. Kim, and H.-J. Yoo, “LightRot: A Lightweight Rotation Scheme and Architecture for Accurate Low-bit Large Language Model Inference,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2025.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.
- [17] Z.-G. Liu, P. N. Whatmough, and M. Mattina, “Systolic tensor array: An efficient structured-sparse gemm accelerator for mobile cnn inference,” *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 34–37, 2020.
- [18] T. Wolf *et al.*, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2020.
- [19] R. Wightman, H. Touvron, and H. Jegou, “ResNet Strikes Back: An Improved Training Procedure in Timm,” in *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021.
- [20] D. Qin *et al.*, “Mobilenetv4: Universal models for the mobile ecosystem,” in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2025, pp. 78–96.
- [21] H. Touvron *et al.*, “Training Data-efficient Image Transformers & Distillation through Attention,” in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 10 347–10 357.
- [22] Z. Liu *et al.*, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 012–10 022.
- [23] H. Cai, J. Li, M. Hu, C. Gan, and S. Han, “EfficientViT: Lightweight Multi-scale Attention for High-resolution Dense Prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 302–17 313.
- [24] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, “FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 5785–5795.
- [25] L. Gao *et al.*, “A Framework for Few-shot Language Model Evaluation,” 07 2024. [Online]. Available: <https://zenodo.org/records/12608602>
- [26] A. Dubey *et al.*, “The Llama 3 Herd of Models,” *arXiv:2407.21783*, 2024.
- [27] L. Wiedmann, A. R. Gosthipaty, and A. Marafioti, “nanovlm,” <https://github.com/huggingface/nanoVLM>, 2025.
- [28] L. Chen, J. Li, X. Dong, P. Zhang, Y. Zang, Z. Chen, H. Duan, J. Wang, Y. Qiao, D. Lin, and F. Zhao, “Are We on the Right Way for Evaluating Large Vision-Language Models?” in *Proceedings of The Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024. [Online]. Available: <https://openreview.net/forum?id=evP9mxNNxJ>
- [29] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, “Sigmoid Loss for Language Image Pre-Training,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 11 975–11 986.
- [30] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlicek, A. P. Lajarin, V. Srivastav, J. Lochner, C. Fahlgrén, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf, “Smollm2: When smol goes big – data-centric training of a small language model,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.02737>
- [31] Y. Chen, A. F. AbouElhamayed, X. Dai, Y. Wang, M. Andronic, G. A. Constantinides, and M. S. Abdelfattah, “BitMoD: Bit-serial Mixture-of-Datatype LLM Acceleration,” in *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1082–1097.