How Far Can MXFP4 Go in Pretraining?

Anonymous authors
Paper under double-blind review

Abstract

While pretraining large language models in 4-bit floating point (FP4) can substantially reduce training cost, FP4 formats struggle to cover the dynamic range of the weights, activations, and gradients encountered during training. Popular FP4 formats therefore pair each small block of values with a shared scale. MXFP4 uses power-of-two block scales that are simpler to support in hardware, while NVFP4 spends mantissa bits on its local scale to reduce quantization error. The two formats thus trade hardware simplicity against scale precision, and recent work reports larger pretraining degradation for MXFP4 than for NVFP4. In this work, we propose a MXFP4 pretraining recipe, MXFP4-MARS, based on recent advanced techniques, including Hadamard rotation, stochastic rounding, and Overflow-Aware Scaling (OAS). Our recipe also leverages Macro Block Scaling (MBS), which maintains higher-precision scale at a coarser granularity while keeping hardware-friendly power-of-two per-block scales. At matched perplexity, our MXFP4 recipe closes the token-overhead gap to NVFP4 from the previously reported 36% down to 0.21% on average, a 171$\times$ reduction in additional tokens required. Also, at the same training budget, a 7B model trained using MXFP4-MARS yields a 1.4% higher average downstream score than the model trained with NVFP4. These results identify scale hierarchy as a key design dimension for low-precision training and show that MXFP4 can be quality-competitive with NVFP4 while keeping the hardware-friendly power-of-two scales, making it an attractive, portable target for FP4 pretraining.

1 Introduction

As pretraining a large language model can consume millions of GPU-hours (Grattafiori et al., 2024), using low-precision training exploiting efficient low-precision tensor cores has gained a lot of attention to accelerate the training. Using FP8 during pretraining is now established at scale (Micikevicius et al., 2022; DeepSeek-AI et al., 2025), and with native FP4 tensor cores available in commercial GPUs (NVIDIA Corporation, 2025; AMD, 2025), recent work is actively pushing precision to four bits (NVIDIA et al., 2026; Tseng et al., 2025).

There are two mainstream FP4 formats that are available on the commercial GPUs. MXFP4 (Open Compute Project, 2023), from the OCP Microscaling specification, pairs E2M1 elements with a single power-of-two E8M0 scale shared across a block. NVFP4 (Alvarez et al., 2025), introduced later by NVIDIA, pairs the same E2M1 elements with an E4M3 block scale and adds an extra per-tensor scale to adjust the range. Unlike NVFP4, which is only supported by NVIDIA GPUs (NVIDIA Corporation, 2025), MXFP4 is supported by multiple vendors, not only NVIDIA Blackwell GPUs (NVIDIA Corporation, 2025), but also AMD MI350 (AMD, 2025), OpenAI Jalapeno (OpenAI, 2026), Microsoft MAIA 200 (Xu et al., 2026), and Meta MTIA 300 (Team, 2026), making MXFP4 the more portable target thanks to its power-of-two scaling that requires less tensor core area (Chhugani et al., 2026).

Early FP4 pretraining studies primarily focused on MXFP4 and explored techniques for mitigating its quantization error (Tseng et al., 2025; Yang et al., 2025; Castro et al., 2025). After NVFP4 was introduced, another line of research has been conducted on pretraining using NVFP4 leveraging its higher-precision scaling (Chmiel et al., 2025; Chen et al., 2026b; Panferov et al., 2026; Cook et al., 2025). A recent study shows that MXFP4 requires 36% more training tokens than NVFP4 to reach the same final loss (NVIDIA et al., 2026). As MXFP4 is the more broadly available format across different HW vendors, closing this gap would make portable FP4 pretraining practical.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A more expressive scale hierarchy is needed without changing MXFP4's standard local block scales to maintain hardware-friendliness. A recent work (Chhugani et al., 2026) introduces Macro Block Scaling (MBS) to address this limitation by maintaining higher-precision scale at a coarser granularity, adding fractional precision above the local E8M0 scales. Because MBS is shared across multiple local blocks, MBS avoids the cost of adding fractional bits to every local scale. They show the potential of the technique on inference workloads, but they have not explored it for training.

Effective FP4 pretraining depends on more than the underlying numerical format. State-of-the-art recipes therefore supplement FP4 with techniques such as stochastic rounding, Hadamard rotation, and 2D block scaling (NVIDIA et al., 2026). We re-evaluate these techniques for MXFP4 pretraining, measuring how each behaves once MBS is in place, and combine the components that hold up (MBS, overflow-aware scaling, Hadamard rotation for WGRAD, and stochastic rounding) into our MXFP4 pretraining recipe, MXFP4-MARS. We apply our recipe to every linear layer within each Transformer block, covering all three GEMM operations in both forward and backward computation. At 7B scale, MXFP4-MARS closely tracks NVFP4 throughout training, requiring only 0.11% to 0.32% more tokens to reach matched validation perplexities. After 640B tokens, it recovers 80% of the downstream score gap between the MXFP4-OCP baseline and BF16. It also achieves 1.4% higher average accuracy than the NVFP4 baseline across benchmarks spanning multiple domains.

Our contributions are:

- **Holistic analysis of the MXFP4 pretraining design space.** We jointly evaluate block size, scale hierarchy, rounding mode, Hadamard rotation, and 1D versus 2D scaling, with all linear layers in FP4. Block size and scale selection account for most of the gap between MXFP4 and NVFP4. MBS requires stochastic rounding; without it, gradient error is decreased by 46% while raising gradient bias by 2.6×, thus increasing final loss. Hadamard rotation helps only in weight gradient, and 2D scaling degrades both formats.
- **A combined MXFP4 recipe, MXFP4-MARS.** Our recipe combines OAS, MBS, gradient stochastic rounding, and weight gradient Hadamard rotation, preserving E2M1 values and power-of-two E8M0 local scales. We also select the macro-block size on quality, metadata, and measured kernel throughput.
- **End-to-end training validation at 7B parameters with 640B tokens.** MXFP4-MARS attains matched validation perplexity with only 0.21% more training tokens than NVFP4 on average. On six downstream benchmarks, it recovers 80% of the difference between the MXFP4-OCP baseline and BF16 while outperforming NVFP4 by 1.4% on average. Component ablations at 7B confirm that each element of the recipe contributes at this scale.

2 Background and Related Work

FP4 formats. FP4 format design has been explored in both academia (Hu et al., 2026b;a; Chen et al., 2026a) and industry (Open Compute Project, 2023; Alvarez et al., 2025; Luo et al., 2026). In this work, we focus on MXFP4 because it is widely supported by AI accelerators (NVIDIA Corporation, 2025; AMD, 2025; OpenAI, 2026; Xu et al., 2026; Team, 2026). MXFP4 assigns one power-of-two E8M0 scale to each group of 32 values, whereas NVFP4 assigns an E4M3 scale to each group of 16 values and adds an extra per-tensor scale. Both use E2M1 elements but organize their scale metadata differently. The two formats therefore differ in block size, scale precision, and scale hierarchy.

Low-bit quantization methods. The limited representational capacity of low-bit formats has motivated a broad range of techniques for reducing quantization error. Some methods combine finer-grained blocks with more expressive scale formats such as UE5M3 (Fasoli et al., 2026; Hu et al., 2026a), while others augment block scales with auxiliary metadata (Hu et al., 2026b; Chen et al., 2026a; Chhugani et al., 2026). Given a target representation, OAS (Chhugani et al., 2026; Yang et al., 2025), Half-S (Du et al., 2026), and Four Over Six (Cook et al., 2025) improve scale selection. GPTQ (Frantar et al., 2023) and OmniQuant (Shao et al., 2024) use calibration data to minimize differences between full-precision and quantized outputs. Other methods make tensor distributions more amenable to quantization through channel-wise smoothing (Xiao et al., 2023), activation-aware scaling (Lin et al., 2024), or orthogonal rotations (Ashkboos et al., 2024; Liu et al., 2025). Finally,

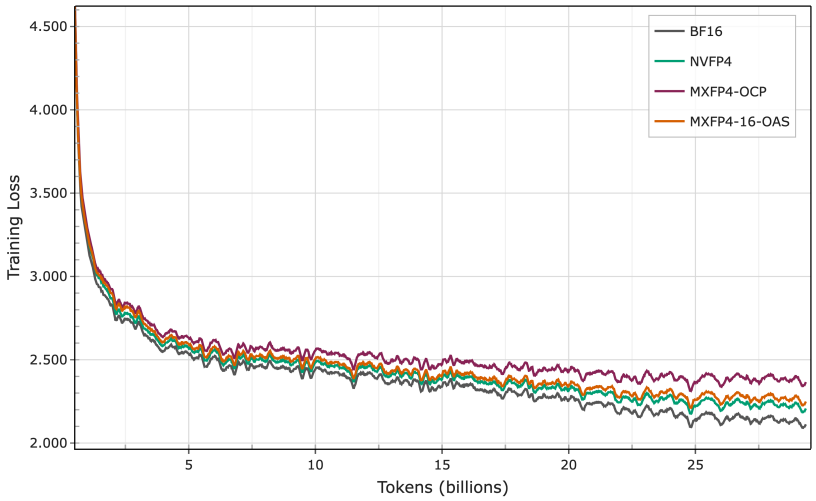


Figure 1: Training loss for a 300M model trained on 30B tokens. MXFP4-OCP uses the default 1×32 blocks without OAS; MXFP4-16-OAS uses 1×16 blocks with OAS.

mixed-precision methods assign higher precision to selected values or model components (Dettmers et al., 2022; 2024; Jeong et al., 2024; Li et al., 2026).

FP4 pretraining recipes. In pretraining, quantization is repeatedly applied to evolving weights, activations, and gradients, causing its errors to propagate through both computation and parameter updates. Existing recipes make tensors easier to quantize through Hadamard rotations (Tseng et al., 2025; Castro et al., 2025; Chen et al., 2026b; Kim et al., 2026; Cim et al., 2026), and reduce gradient bias through stochastic rounding (Tseng et al., 2025; Yang et al., 2025; Chmiel et al., 2025) or alternative unbiased estimators (Panferov et al., 2026). They also retain higher precision for selected computations (Zhou et al., 2025; Cao et al., 2026; Pan et al., 2026) or vary which operations use FP4 (Cim et al., 2026; Ding et al., 2026). Large-scale FP4 pretraining typically relies on a combination of these techniques. NVIDIA trains a 12B hybrid Mamba-Transformer on 10T tokens with selective high-precision layers, Hadamard rotations, two-dimensional (2D) weight scaling, and stochastic rounding (NVIDIA et al., 2026). A scaling study of 8B dense and 30B MoE models on up to 1T tokens also uses selective high-precision layers and Hadamard rotations, but adopts one-dimensional weight scaling with backward re-quantization, adaptive scale selection, and an improved unbiased estimator (Agrusa et al., 2026). Our work systematically evaluates stochastic rounding, Hadamard rotation, and 2D scaling while quantizing all linear layers without high-precision fallbacks.

3 Closing the Scale Gap

3.1 MXFP4-OCP and MXFP4-16-OAS

MXFP4 differs from NVFP4 in block size, scale precision, and scale hierarchy. We first align their block sizes by reducing the MXFP4 block size from the OCP default of 1×32 to 1×16 to understand the effectiveness of MXFP4. With fewer values sharing each scale, each block covers a narrower dynamic range, allowing its scale to better preserve small values. We observe that reducing the block size lowers the average flush-to-zero rate by about 3 percentage points and improves Quantization Signal-to-Noise Ratio (QSNR) by up to 0.51 dB.

We next refine the choice of the power-of-two scale with Overflow-Aware Scaling (OAS) (Chhugani et al., 2026). The OCP specification selects an E8M0 scale $d = 2^e$ such that the normalized block maximum $\alpha_{\max}/d$ lies in $[4, 8)$, which can exceed E2M1's maximum of 6 and incur large saturation error. Using the next larger scale, 2^{e+1}, avoids saturation but rounds more small values to zero, potentially suppressing training gradients. OAS chooses between 2^e and 2^{e+1}, retaining OCP's 2^e only when the block maximum's saturation error under 2^e is no greater than its quantization error

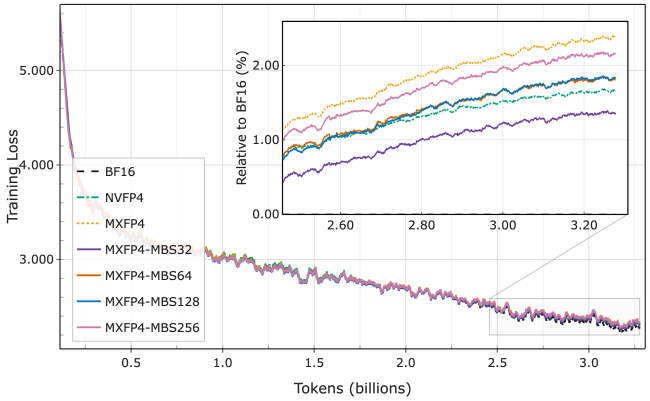


Figure 2: Training loss across MBS sizes on a 300M model trained for 3.28B tokens. MXFP4 denotes MXFP4-16-OAS. The inset shows the training loss increase relative to BF16 during the final 0.8B tokens. Lower is better.

Table 1: Final loss gap relative to BF16 and scale metadata for the 300M model.

Format	Loss gap (%)	Metadata (bits/elem.)
NVFP4	1.66	0.5000*
MXFP4	2.39	0.5000
MBS32	1.36	0.7500
MBS64	1.82	0.6250
MBS128	**1.84**	**0.5625**
MBS256	2.17	0.5313

Bold denotes the selected MBS128 configuration; * NVFP4's per-tensor scale is excluded because its cost depends on tensor shape.

under 2^{e+1}. We use MXFP4-16-OAS to represent MXFP4 with a block size of 16 and OAS, and MXFP4-OCP to represent the OCP configuration with a block size of 32. We observe OAS improves average QSNR by 0.66 dB. Figure 1 reports training loss for MXFP4-16-OAS on a 300M model trained for 30B tokens. MXFP4-16-OAS substantially reduces the gap to NVFP4, with the relative final loss difference decreasing from 7.18% for MXFP4-OCP to 1.89%. Appendix C further analyzes the individual contributions of block-size reduction and OAS.

3.2 Macro Block Scaling

The smaller block size and OAS reduce the gap to NVFP4 but do not eliminate it (Figure 1). The remaining gap stems from the limited precision of MXFP4's exponent-only E8M0 scale. We address this limitation with Macro Block Scaling (MBS) (Chhugani et al., 2026), which groups multiple consecutive 1×16 blocks into a macro-block that shares a single 8-bit mantissa m. The shared mantissa provides finer scale resolution than E8M0 alone.

To construct this two-level scale, MBS computes the full-precision quantization factor $q_{\mathrm{M}} = \frac{\alpha_{\max}}{6}$ from the macro-block absolute maximum $\alpha_{\max}$. Then we decompose this factor into a power-of-two component and a mantissa component, $q_{\mathrm{M}} = 2^r(1 + \widetilde{m})$. MBS stores the eight most significant bits of $\widetilde{m}$ as m, forming the shared factor $c = 1 + m \in [1, 2)$. Each value is then divided by c before local E8M0 quantization. For local block b, OAS selects an E8M0 scale $d_b = 2^{e_b}$ from its scaled maximum $\frac{\alpha_{\max,b}}{c}$.

The power-of-two component is encoded in the local scales d_b, so only the shared mantissa m adds macro-block metadata. This max-based procedure is static MBS. Dynamic MBS instead searches over candidate shared mantissas to reduce quantization error and potentially improve quantization accuracy, but adds substantial overhead at every training step. We therefore use static MBS throughout pretraining.

We apply MBS during MXFP4 quantization of the linear layers within each Transformer block in both the forward and backward passes. For each operand, we align the local E8M0 blocks and MBS macro-blocks along the reduction dimension K. This alignment allows us to apply the inverse MBS factors of both operands to each macro-block's partial result before accumulation across K, while preserving hardware acceleration for the underlying FP4 multiplication.

3.3 Macro Block Size Trade-offs

The macro-block size N determines how many consecutive 1×16 MXFP4 blocks share one mantissa. We choose N by balancing scale metadata cost, training quality, and runtime overhead. A smaller N allows each mantissa to better match local value ranges, improving quantization fidelity, whereas a larger N amortizes the shared mantissa and its associated rescaling overhead across more elements.

Table 1 reports the metadata cost for $N \in \{32, 64, 128, 256\}$. Each local E8M0 scale contributes $8/16 = 0.5$ bits per element, and the shared 8-bit mantissa contributes $8/N$, giving a total of $0.5 + 8/N$ bits per element. As N increases, the scale metadata cost decreases from 50% above that of MXFP4 for MBS32 to just 6.3% above it for MBS256, but these savings can come at the expense of training quality.

To measure this quality trade-off, we sweep the macro-block size on a 300M model. Figure 2 shows the resulting training loss curves. The MXFP4 baseline has a 2.39% loss gap relative to BF16, 44% larger than NVFP4's. MBS256 increases scale metadata by only 6.3% but closes about one third of the gap between MXFP4 and NVFP4. MBS128 yields a much larger improvement, closing 75% of this gap with 12.5% metadata overhead and reducing the loss gap to 1.84%. Reducing N further from 128 to 64 doubles the metadata overhead but lowers the loss gap only slightly. MBS32 achieves lower loss than NVFP4 but requires 50% more scale metadata.

The quality gains from reducing the macro-block size also come at a runtime cost. In our implementation, we fuse the inverse scaling required by MBS into GEMM. For each macro-block along K, the kernel computes a partial result, undoes the shared scaling applied to both operands, and adds the corrected result to the output accumulator. The underlying block-scaled FP4 matrix multiply-accumulate (MMA) supported on NVIDIA Blackwell GPUs spans 64 elements along K per operation. This means that MBS32 requires two separately rescaled partial results within a single MMA range, forcing the $K = 64$ MMA to be repeated over the same elements. Given this overhead, our runtime analysis considers only MBS64 and larger configurations, whose corrections align with one or more complete MMA operations and avoid this redundancy. In our FP4 GEMM benchmarks reported in Appendix D, increasing N from 64 to 128 improves effective TFLOPS by 15.8%.

Taken together, these results identify MBS128 as the knee of the quality–metadata–runtime trade-off. Relative to MBS64, MBS128 maintains comparable training quality while halving the additional scale metadata and reducing runtime overhead. Increasing N beyond 128 further reduces metadata and runtime overhead but degrades training quality. We therefore select **MBS128** for all subsequent experiments.

4 Building the MXFP4 Training Recipe

Section 3 establishes an MXFP4 configuration with 1×16 blocks and OAS, then adds MBS128. Pretraining involves repeatedly quantizing evolving weights, activations, and gradients, so that each step's quantization error feeds back into subsequent updates and can accumulate over training. Closing the scale gap alone may therefore be insufficient to preserve training quality. We evaluate how MBS interacts with established FP4 training techniques. Specifically, we examine stochastic rounding, Hadamard rotation, and 2D block scaling, all used in NVIDIA's NVFP4 pretraining recipe (NVIDIA et al., 2026). We select the final recipe through ablations on a 300M-parameter model.

Our recipe applies FP4 to every linear layer within each Transformer block during both the forward and backward passes, without high-precision (BF16) fallbacks. The previous recipe (NVIDIA et al., 2026) keeps 15% of linear layers in BF16; at the 3.3× FP4 GEMM speedup we measure (Table 5), keeping 15% of linear-layer FLOPs in BF16 would consume 37% of linear-layer time, equivalent to running half the model in FP4. We also aim to characterize how far MXFP4 alone can go rather than to match BF16 with mixed precision; selective high precision remains complementary (Section 6).

4.1 Selecting Recipe Components

Stochastic rounding (SR). Gradients are particularly vulnerable to FP4 quantization because many values are small relative to the block maximum. Under round-to-nearest-even (RTNE), block-scaled values below the midpoint between zero and the smallest nonzero E2M1 value are deterministically rounded to zero, diminishing the gradient signal and hindering optimization. SR instead gives these small values a chance to round away from zero, preserving their contribution in expectation. Prior work has shown SR to be effective for FP4 training, and it has become a standard choice in recent recipes (Tseng et al., 2025; Yang et al., 2025; Chmiel et al., 2025; NVIDIA et al., 2026). We evaluate the effect of rounding mode by comparing RTNE and SR across MXFP4, MXFP4-MBS128, and NVFP4. We use MXFP4-16-OAS as MXFP4 in this experiment.

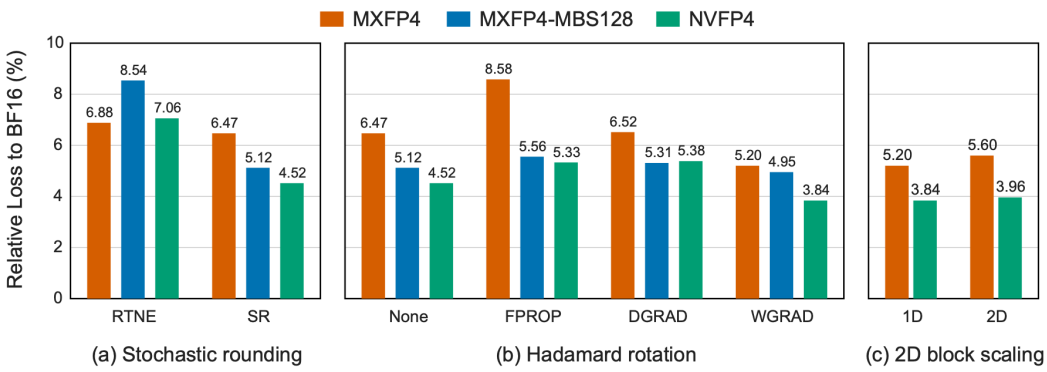


Figure 3: Recipe ablations on a model with 300M parameters. Bars show the increase in training loss relative to BF16. Lower is better. We compare (a) RTNE and SR for MXFP4, MXFP4-MBS128, and NVFP4, (b) Hadamard rotation applied separately in FPROP, DGRAD, and WGRAD, using no rotation as the baseline, and (c) 1D and 2D block scaling for MXFP4 and NVFP4 with WGRAD Hadamard rotation.

Figure 3(a) shows that all three configurations benefit from SR: the final loss gap relative to BF16 decreases from 6.88% to 6.47% for MXFP4, from 8.54% to 5.12% for MXFP4-MBS128, and from 7.06% to 4.52% for NVFP4. The results also indicate that MBS128 can increase the loss gap under RTNE. Despite lowering quantization error, MBS128 produces a larger-magnitude running mean bias in gradient quantization under RTNE, which could hinder optimization. Appendix E shows how SR mitigates this effect. Accordingly, we use SR for gradient quantization throughout the recipe.

Hadamard rotation. Outliers can dominate a shared block scale and reduce the effective precision of smaller values. Hadamard rotation mitigates this effect by spreading outlier energy before quantization (Tseng et al., 2025; Castro et al., 2025; Kim et al., 2026). To determine where rotation is most effective, we evaluate a randomized size-16 transform, matching the local FP4 block size, separately in the forward (FPROP), input-gradient (DGRAD), and weight-gradient (WGRAD) GEMMs. Figure 3(b) shows that Hadamard rotation reduces the loss gap when applied to WGRAD, but provides no benefit in FPROP or DGRAD. This pattern holds for MXFP4 with or without MBS128 and for NVFP4. WGRAD benefits most because it operates on both outlier-prone activations and gradients, whereas FPROP and DGRAD each pair one of them with weights that have fewer outliers. At this scale, NVFP4 with WGRAD rotation achieves the lowest loss gap of 3.84%, while MBS128 with WGRAD rotation reaches 4.95%, the best MXFP4 result. Appendix F further shows that the advantage of WGRAD rotation with MBS128 becomes more pronounced toward the end of longer training. We therefore include WGRAD Hadamard rotation in the final recipe.

2D block scaling. With one-dimensional (1D) block scaling, transposing a tensor changes its scale groups and thus its quantized representation. For weights, this creates a mismatch between the representation used to produce activations and the one used to propagate their gradients. Prior work (NVIDIA et al., 2026) addresses this mismatch with two-dimensional (2D) scaling, assigning one scale to each 16×16 tile so that the quantized representation is preserved under transposition. However, this consistency comes at the cost of expanding each scaling group from 16 to 256 values, which can increase quantization error. We evaluate whether this consistency benefit outweighs the quantization cost of larger scaling groups for MXFP4 and NVFP4 with WGRAD Hadamard rotation. Figure 3(c) shows that 2D scaling instead increases the BF16-relative loss gap by 7.64% for MXFP4 and 3.21% for NVFP4. The 2D layout also complicates MBS; grouping tiles along one axis would reintroduce orientation dependence, while a 2D macro-tile would require a new metadata layout and correction scheme. Given the higher loss gap and this added complexity, we use 1D scaling.

4.2 Final Recipe, MXFP4-MARS

Combining these choices gives our final recipe, **MXFP4-MARS** (MXFP4-16-OAS-MBS128-Had-SR), named for **M**acro Block Scaling, Overflow-**A**ware Scaling, Hadamard **R**otation, and **S**tochastic

rounding. It uses 1×16 MXFP4 blocks with OAS and MBS128, grouping eight consecutive local blocks along the reduction dimension into a 128-element macro-block with one shared 8-bit mantissa. Gradients use stochastic rounding, WGRAD uses a randomized Hadamard transform of size 16, and block scaling remains 1D. The pipeline first applies Hadamard rotation where enabled, then applies the MBS shared-mantissa transform, selects local E8M0 scales with OAS, and finally quantizes values to E2M1 using RTNE by default and SR for gradients. We apply FP4 to FPROP, DGRAD, and WGRAD in every linear layer within each Transformer block.

5 MXFP4-MARS PRETRAINING

5.1 SETUP

Model and training setup. We pretrain a 7B dense Transformer model from scratch for 640B tokens. It follows a Llama 3-style architecture with 32 Transformer blocks and a hidden size of 4096 (Grattafiori et al., 2024). Appendix A provides additional model details. We train with AdamW using $\beta_1 = 0.9$ and $\beta_2 = 0.99$ (Loshchilov & Hutter, 2019). We use the warmup-stable-decay (WSD) learning rate schedule (Hu et al., 2024), with peak and minimum learning rates of 4×10^{-3} and 4×10^{-5}. After a 10% warmup, the learning rate remains at its peak until decaying to the minimum over the final 10% of training. Within each experiment, all precision configurations share the same initialization, data order, and training hyperparameters.

Quantization configurations. We quantize weights, activations, and gradients in all Transformer linear layers, including the QKV and output projections in attention and the linear layers in the feed-forward blocks, using fake quantization to emulate the corresponding quantization and dequantization operations. Other operations remain in BF16, while master weights and optimizer states are FP32. We compare MXFP4-MARS with BF16 and two FP4 baselines. MXFP4-MARS is our final recipe, combining MXFP4-16-OAS with MBS128, gradient SR, and WGRAD Hadamard rotation. MXFP4-OCP uses the original 1×32 OCP MXFP4 scaling, while NVFP4 provides the NVFP4 baseline. For a fair comparison, we apply gradient SR and WGRAD Hadamard rotation to both MXFP4 and NVFP4 baselines, matching the corresponding components of MXFP4-MARS. BF16 serves as the higher-precision reference.

Evaluation setup. We evaluate model quality using validation perplexity and downstream accuracy. The perplexity evaluation reports a category-balanced geometric mean over six internal datasets spanning coding, books, and notes. For downstream accuracy, we use six benchmarks: MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), Multilingual MMLU (OpenAI, 2024), GPQA (Rein et al., 2023), GSM8K-Platinum (Vendrow et al., 2025), and MGSM (Shi et al., 2023). The suite covers general and expert knowledge, scientific and mathematical reasoning, and multilingual capabilities.

Table 2: Downstream accuracy (%) at the final checkpoint after 640B tokens of pretraining. Avg is the unweighted mean across the six benchmarks, and Δ is the relative difference from BF16. Bold values mark the best FP4 result in each column. GSM8K-P = GSM8K-Platinum.

Configuration	MMLU	MMLU-Pro	MMMLU	GPQA	GSM8K-P	MGSM	Avg	Δ(%)
BF16	55.56	25.85	36.12	28.16	51.61	39.53	39.47	—
MXFP4-OCP (with Had, SR)	34.67	15.92	30.90	20.45	26.55	20.44	24.82	−37.12
NVFP4 (with Had, SR)	50.41	23.73	35.04	**26.69**	44.33	**35.60**	35.97	−8.88
MXFP4-MARS	**52.48**	**24.68**	**35.10**	26.53	**45.16**	34.95	**36.48**	**−7.57**

5.2 DOWNSTREAM EVALUATION

We first compare downstream performance after 640B tokens of pretraining. As Table 2 shows, MXFP4-OCP achieves an average score of 24.82%, which is 37.1% below BF16. MXFP4-MARS closes most of this gap, scoring only 7.6% below BF16 and recovering 80% of the difference between the MXFP4-OCP baseline and BF16. Against NVFP4, which remains 8.9% below BF16, MXFP4-MARS delivers 1.4% higher average accuracy and leads on four of the six benchmarks. We also report a broader set of precision configurations trained with a cosine learning rate in Appendix B.

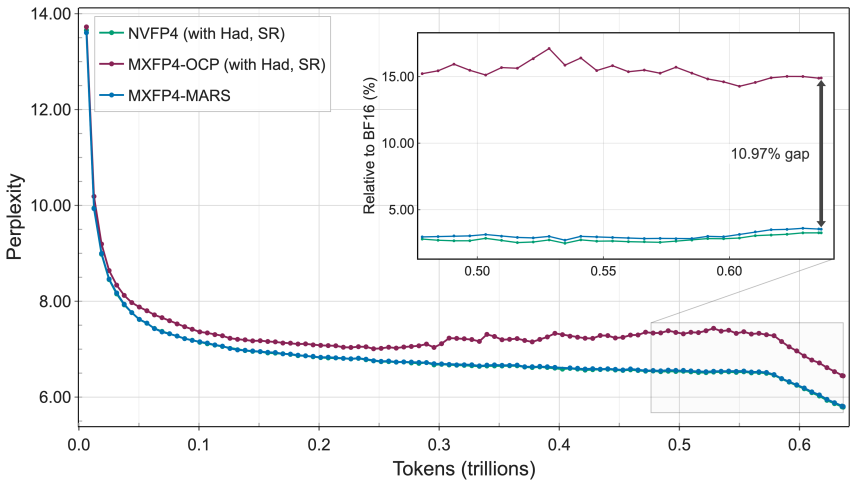


Figure 4: Validation perplexity throughout training for MXFP4-OCP, NVFP4, and MXFP4-MARS. All configurations use gradient SR and WGRAD Hadamard rotation. The inset shows the relative perplexity gap to BF16 late in training and highlights that MXFP4-OCP has 10.97% higher perplexity than MXFP4-MARS at the final checkpoint.

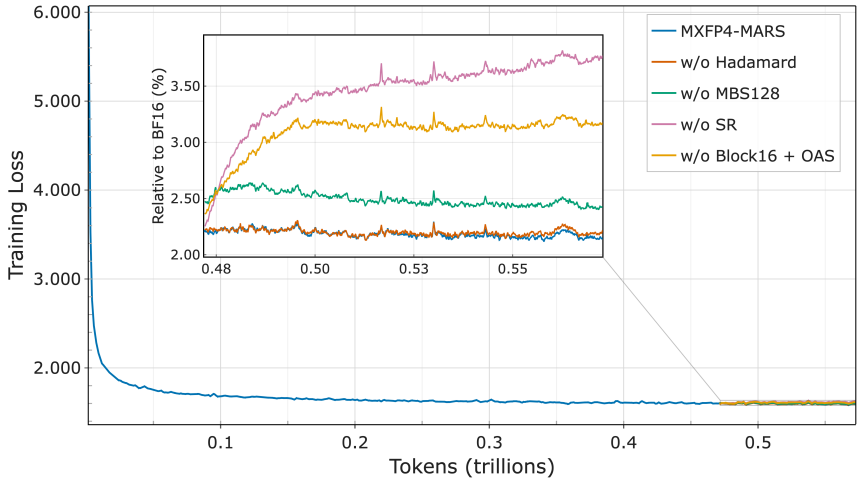


Figure 5: Recipe component ablations at 7B. Each variant resumes from MXFP4-MARS after 472B tokens with one component removed: WGRAD Hadamard rotation, MBS128, gradient SR, or Block16+OAS. The inset shows training loss relative to BF16 after branching from the shared checkpoint.

5.3 Perplexity and Token Efficiency

We next compare validation perplexity and token efficiency under the WSD schedule for NVFP4, MXFP4-OCP, and MXFP4-MARS. Figure 4 shows that MXFP4-OCP departs from NVFP4 early in training, while MXFP4-MARS closely tracks it. At the final checkpoint, MXFP4-OCP has a 14.9% relative perplexity gap to BF16. MXFP4-MARS narrows the gap to 3.54%, only 0.27 percentage points above the 3.27% gap for NVFP4.

We also compare token efficiency at matched validation perplexities. We define target perplexities from 6.44 to 5.80 at intervals of 0.02, starting at the lowest perplexity achieved by MXFP4-OCP and ending at the lowest achieved by NVFP4. For each target reached by MXFP4-MARS, we compare its required token count with that of NVFP4. Across these targets, MXFP4-MARS requires only 0.11% to 0.32% more training tokens than NVFP4, with a mean overhead of 0.21%.

5.4 Recipe Component Ablations

We next evaluate the contribution of each recipe component at 7B. For each ablation, we resume training from MXFP4-MARS after 75% of the full training (472B tokens). We separately disable gradient SR, MBS128, and Hadamard rotation, and replace Block16+OAS with MXFP4-OCP scaling.

Figure 5 shows that removing any component increases training loss relative to MXFP4-MARS. Disabling gradient SR has the largest effect, ending with a loss 3.74% above BF16. Its gap from MXFP4-MARS continues to widen. Reverting Block16+OAS to MXFP4-OCP scaling produces the next largest degradation, with a final gap of 3.17%. Without MBS128, training loss is consistently higher than with MXFP4-MARS, ending 2.42% above BF16. The sustained gap demonstrates the benefit of MBS128 throughout the run. Removing Hadamard rotation results in a final gap of 2.19%, close to that of MXFP4-MARS. The small difference arises because Hadamard is disabled only after 75% of the training, not because its overall benefit is limited. Over the full training horizon, Hadamard's benefit becomes increasingly pronounced, as shown in Appendix F.

6 Discussion

Combining MBS with other techniques. In this work, we show that MBS is not only effective for inference, but also compatible with OAS, gradient SR, and WGRAD Hadamard rotation during training. Beyond the placements evaluated here, other variants of Hadamard rotation (Ashkboos et al., 2025; Kim et al., 2026) can interact differently with MBS, motivating systematic evaluation of these combinations. For gradient quantization, MS-EDEN (Panferov et al., 2026) provides an alternative to SR, and its combination with MBS128 in MXFP4 training remains to be explored. For scale selection, adaptive methods such as Four Over Six (Cook et al., 2025) and dynamic MBS (Chhugani et al., 2026) could be evaluated within our pretraining recipe.

Scope and remaining gap. Our experiments focus on dense Llama 3-style models with up to 7B parameters and do not cover other architectures such as mixture-of-experts models. Unlike approaches that use BF16 or FP8 for selected layers or outlier values (NVIDIA et al., 2026; Agrusa et al., 2026; Chen et al., 2026b; Pan et al., 2026), we quantize every Transformer linear layer in both FP4 formats. In this setting, our recipe largely closes the MXFP4–NVFP4 gap but remains below BF16. Combining MBS with such selective high-precision computation would further narrow the remaining gap to BF16.

7 Conclusion

We present MXFP4-MARS, a pretraining recipe for MXFP4 that combines 1×16 blocks and OAS with MBS128, gradient SR, and WGRAD Hadamard rotation. The recipe quantizes every Transformer linear layer while retaining standard E2M1 values and power-of-two E8M0 local scales. Our analysis identifies scale hierarchy as a key source of the quality gap between MXFP4 and NVFP4 and shows how MBS narrows this gap without changing MXFP4's hardware-friendly local representation. On a 7B model pretrained for 640B tokens, MXFP4-MARS requires only 0.21% more training tokens than NVFP4 on average at matched validation perplexities. It also recovers 80% of the downstream score gap between the MXFP4-OCP baseline and BF16 and achieves a 1.4% higher average score than NVFP4 across six benchmarks. These results show that MXFP4 can be a quality-competitive and portable target for FP4 pretraining.

AI Use Statement

We used generative AI tools to assist with translation, quantitative consistency checks, drafting and editing, literature search and reference formatting, and figure generation and formatting. We have not used generative AI tools to design or implement research methods or experiments; other required-disclosure tasks are not applicable to this work. We developed all methods, produced all underlying data, and reviewed and verified all AI-assisted content. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Anjulie Agrusa, Andrei Panferov, Elizabeth Wei, Keith Wyss, Paul Gibbons, Erik Schultheis, Tijmen Blankevoort, and Dan Alistarh. What matters for NVFP4 training? a scaling study of low-precision pre-training recipes. In *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026. URL https://openreview.net/forum?id=jlkIyaG32w.

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 2025. URL https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/.

AMD. AMD Instinct MI350X GPUs. AMD Product Page, 2025. URL https://www.amd.com/en/products/accelerators/instinct/mi350/mi350x.html. Launch date: June 12, 2025. CDNA4 architecture with native MXFP4 and MXFP6 support, 9.2 PFLOPs peak MXFP4 performance, 288 GB HBM3E.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems 37*, pp. 100213–100240. Neural Information Processing Systems Foundation, Inc., 2024. doi: 10.52202/079017-3180.

Saleh Ashkboos, Mahdi Nikdan, Rush Tabesh, Roberto Castro, Torsten Hoefler, and Dan Alistarh. HALO: Hadamard-assisted lower-precision optimization for LLMs. In *Advances in Neural Information Processing Systems 38*, pp. 131755–131780. Neural Information Processing Systems Foundation, Inc., 2025. doi: 10.52202/085713-3966.

Hengjie Cao, Mengyi Chen, Yifeng Yang, Ruijun Huang, Fang Dong, Jixian Zhou, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, Yuan Cheng, Fan Wu, Fan Yang, Tun Lu, Ning Gu, and Li Shang. Metis: Training LLMs with FP4 quantization. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.00404.

Roberto Castro, Andrei Panferov, Rush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native FP4 training can be optimal for large language models. In *Advances in Neural Information Processing Systems 38*, pp. 48625–48645. Neural Information Processing Systems Foundation, Inc., 2025. doi: 10.52202/085713-1449.

Chun-Ting Chen, Dongmin Han, Hangyeol Mun, Jake Hyun, Arnab Raha, Amit Agarwal, Mark Anders, Mohamed Abdelfattah, and Jae-sun Seo. HBQ: Hierarchical scaling block quantization with hardware-efficiency-aware design for accurate LLM inference. In *59th IEEE/ACM International Symposium on Microarchitecture*, 2026a. URL https://arxiv.org/abs/2609.00450.

Yuxiang Chen, Yifan Liu, Xiaoming Xu, Pengle Zhang, Michael Beyer, Martin Rapp, Jun Zhu, and Jianfei Chen. TetraJet-v2: Accurate NVFP4 training for large language models with oscillation suppression and outlier control. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026b. URL https://arxiv.org/abs/2510.27527.

Jatin Chhugani, Geonhwa Jeong, Bor-Yiing Su, Yunjie Pan, Hanmei Yang, Aayush Ankit, Jiecao Yu, Summer Deng, Yunqing Chen, Nadathur Satish, and Changkyu Kim. Unveiling the potential of quantization with MXFP4: Strategies for quantization error reduction, 2026. URL https://arxiv.org/abs/2603.08713.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. Fp4 all the way: Fully quantized training of large language models. In *Advances in Neural Information Processing Systems 38*, pp. 101214–101235. Neural Information Processing Systems Foundation, Inc., 2025. doi: 10.52202/085713-3049.

Musa Cim, Sarthak Arora, Poovaiah Palangappa, Miro Hodak, Ravi Dwivedula, Meena Arunachalam, and Mahmut Taylan Kandemir. Pretraining large language models with MXFP4 on native FP4 hardware, 2026. URL https://arxiv.org/abs/2605.09825.

Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, Keith Wyss, Mahdi Nazemi, Asit Mishra, Carlo del Mundo, Tijmen Blankevoort, and Song Han. Four over six: More accurate NVFP4 quantization with adaptive block scaling, 2025. URL https://arxiv.org/abs/2512.02010.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. DeepSeek-V3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35*, pp. 30318–30332. Neural Information Processing Systems Foundation, Inc., 2022. doi: 10.52202/068431-2198.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Q1u25ahSuy.

Siyu Ding, Mingchuan Ma, Jiabo Tong, Xingrun Xing, Ziming Wang, and Guoqi Li. Full-stack FP4: Stable LLM pretraining with quantized projections, optimizers, and attention, 2026. URL https://arxiv.org/abs/2607.04422.

Jinyang Du, Ruihao Gong, Linghan Ai, Zining Wang, Yunke Peng, Yao Wang, Lei Yan, Xuefei Wang, Yaoyuan Wang, Jinyang Guo, Dahua Lin, and Xianglong Liu. Half-s: Halving the scale for near-lossless 4-bit LLM training. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4890–4903, 2026. doi: 10.18653/v1/2026.findings-acl.241.

Andrea Fasoli, Monodeep Kar, Chi-Chun Liu, Swagath Venkataramani, Viji Srinivasan, Leland Chang, and Naigang Wang. Is finer better? the limits of microscaling formats in large language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.19026.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James

Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Robert Hu, Carlo Luschi, and Paul Balanca. UE5M3 FP4 block scaling for stable language model pretraining, 2026a. URL https://arxiv.org/abs/2609.02846.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Weiming Hu, Zihan Zhang, Haoyan Zhang, Chen Zhang, Cong Guo, Yu Feng, Tianchi Hu, Guanglin Li, Guipeng Hu, Junsong Wang, and Jingwen Leng. M2XFP: A metadata-augmented microscaling data format for efficient low-bit quantization. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1151–1167. ACM, 2026b. doi: 10.1145/3779212.3790185.

Geonhwa Jeong, Po-An Tsai, Stephen W. Keckler, and Tushar Krishna. Sdq: Sparse decomposed quantization for llm inference, 2024. URL https://arxiv.org/abs/2406.13868.

Seonggon Kim, Alireza Khodamoradi, Pranathi Vasireddy, Kristof Denolf, and Eunhyeok Park. AdaHOP: Fast and accurate low-precision training via outlier-pattern-aware rotation, 2026. URL https://arxiv.org/abs/2604.02525.

Xinlin Li, Timothy Chou, Josh Fromm, Zichang Liu, Yunjie Pan, and Christina Fragouli. ScaleBITS: Scalable bitwidth search for hardware-aligned mixed-precision LLMs, 2026. URL https://arxiv.org/abs/2602.17698.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

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

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yuanyong Luo, Jing Huang, Yu Cheng, Ziwei Yu, Kaihua Tang, Xinda Ma, Xin Wang, Anping Tong, Guipeng Hu, Yun Xu, Mehran Taghian, Peng Wu, Guanglin Li, Yunke Peng, Tianchi Hu, Minqi Chen, Michael Bi Mi, Hu Liu, Xiping Zhou, Junsong Wang, Qiang Lin, and Heng Liao. HiFloat4 format for language model inference, 2026. URL https://arxiv.org/abs/2602.11287.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 formats for deep learning, 2022. URL https://arxiv.org/abs/2209.05433.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, et al. Pretraining large language models with NVFP4, 2026. URL https://arxiv.org/abs/2509.25149.

NVIDIA Corporation. NVIDIA RTX Blackwell GPU Architecture. White paper, version 1.1, NVIDIA Corporation, 2025. URL https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf.

Open Compute Project. OCP microscaling formats (MX) v1.0 specification. Technical report, Open Compute Project, 2023. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

OpenAI. Multilingual massive multitask language understanding (MMMLU). Hugging Face dataset, 2024. URL https://huggingface.co/datasets/openai/MMMLU.

OpenAI. Jalapeño's first results show industry-leading speed and efficiency in AI inference, 2026. URL https://openai.com/index/jalapeno-first-results/.

Yunjie Pan, Yongyi Yang, Hanmei Yang, and Scott Mahlke. SNIP: An adaptive mixed precision framework for subbyte large language model training. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1815–1831. ACM, 2026. doi: 10.1145/3779212.3790223.

Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet ii: Accurate LLM pre-training in NVFP4 by improved unbiased gradient estimation, 2026. URL https://arxiv.org/abs/2601.22813.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

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

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03057.

MTIA Team. Mtia 300: Meta's first training chip featuring built-in nics and collective offloading engines. In *2026 ACM/IEEE 53rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 1084–1099, 2026. doi: 10.1109/ISCA66397.2026.00085.

Albert Tseng, Tao Yu, and Youngsuk Park. Training LLMs with MXFP4. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1630–1638. PMLR, 2025. URL https://proceedings.mlr.press/v258/tseng25a.html.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. Do large language model benchmarks test reliability?, 2025. URL https://arxiv.org/abs/2502.03461.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37*, pp. 95266–95290. Neural Information Processing Systems Foundation, Inc., 2024. doi: 10.52202/079017-3018.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Sherry Xu, Marco Heddes, Jackson Peng, Tom Savell, Monica Tang, Prashant Ranjan, Jesse Benson, Ofer Dekel, Saurabh Dighe, Anupama Kurpad, Artour Levin, Matthew Mattina, George Petre, Cheng Tang, Yuan Yu, Li Zhang, and Torsten Hoefler. Maia 200: A software defined dataflow system for large-scale ai acceleration, 2026. URL https://arxiv.org/abs/2608.24664.

Hanmei Yang, Summer Deng, Amit Nagpal, Maxim Naumov, Mohammad Janani, Tongping Liu, and Hui Guan. An empirical study of microscaling formats for low-precision LLM training. In *2025 IEEE 32nd Symposium on Computer Arithmetic (ARITH)*, pp. 1–8. IEEE, May 2025. doi: 10.1109/ARITH64983.2025.00011.

Jiecheng Zhou, Ding Tang, Rong Fu, Boni Hu, Haoran Xu, Yi Wang, Zhilin Pei, Zhongling Su, Liang Liu, Xingcheng Zhang, and Weiming Zhang. Towards efficient pre-training: Exploring FP4 precision in large language models, 2025. URL https://arxiv.org/abs/2502.11458.

A Model Configurations

We use a 300M model for the scale and recipe ablations in Sections 3 and 4, and a 7B model for the end-to-end experiments in Section 5. Both are dense Llama 3-style Transformers with SwiGLU activations. Table 3 summarizes their model configurations and training budgets. The 300M MBS-size sweep uses 3.28B tokens, while the other 300M ablations use 30B tokens.

Table 3: Model configurations for the 300M and 7B models.

Model	# of Blocks	Hidden Size	Q heads	KV heads	Sequence Length	Training Tokens
300M	8	1024	8	1	8192	3.28B / 30B
7B	32	4096	32	4	8192	640B

B Downstream Evaluation with Cosine Learning Rate Schedule

In addition to the WSD results in Section 5.2, we evaluate models trained from scratch for 640B tokens using a cosine learning rate schedule. The learning rate warms up linearly to 4×10^{-3} over the first 10% of training and then follows a cosine decay to 4×10^{-5} (Loshchilov & Hutter, 2017).

Table 4: Downstream accuracy (%) under the cosine learning rate schedule at the final checkpoint after 640B tokens of pretraining. Avg is the unweighted mean across the six benchmarks, and Δ is the relative difference from BF16. Bold values mark the best FP4 result in each column. GSM8K-P = GSM8K-Platinum.

Configuration	**MMLU**	**MMLU-Pro**	**MMMLU**	**GPQA**	**GSM8K-P**	**MGSM**	**Avg**	**Δ(%)**
BF16	58.5	27.6	43.8	28.1	51.5	40.0	41.6	—
MXFP6	58.0	28.5	43.8	30.9	52.2	41.4	42.5	+2.1
MXFP4-16-OAS (with SR)	47.6	19.5	35.0	19.9	27.5	24.3	29.0	−30.3
MXFP4-16-OAS (with Had, SR)	53.8	22.9	40.4	26.6	43.7	33.8	36.9	−11.3
MXFP4-16-OAS-MBS128 (with SR)	53.7	23.9	39.8	27.2	45.7	36.0	37.7	−9.3
MXFP4-MARS	54.3	**25.3**	**41.5**	**28.1**	**48.2**	**36.7**	**39.0**	**−6.2**
NVFP4 (with SR)	51.8	20.4	38.3	25.0	34.7	29.0	33.2	−20.2
NVFP4 (with Had, SR)	**54.7**	23.0	40.1	26.6	45.8	35.0	37.5	−9.7

Under the cosine schedule, Table 4 shows how Hadamard rotation and MBS128 improve MXFP4-16-OAS with gradient SR. This baseline is 30.3% below BF16, compared with 20.2% for NVFP4 under the same SR setting, while MXFP6 remains comparable to BF16. Adding Hadamard rotation or MBS128 narrows the gap to 11.3% and 9.3%, respectively. MXFP4-MARS combines both techniques, reducing the gap to 6.2% and recovering 80% of the difference between the MXFP4-16-OAS baseline and BF16. It also delivers 4.0% higher average accuracy than NVFP4 under the same gradient SR and WGRAD Hadamard settings and leads on five of the six benchmarks.

C Effects of Block Size and OAS

To isolate the effects of block size and scale selection, we measure tensor-level QSNR and flush-to-zero (FTZ) rate on 100 tensor samples collected from the linear layers of the 7B model at 630B training tokens, near the end of training. Figure 6 compares six MXFP4 configurations spanning two block sizes, 1×32 and 1×16, and three block-scale selection methods. As described in Section 3.1, OCP uses 2^e, CEIL always takes 2^{e+1}, and OAS chooses between the two scales based on quantization error. OCP and OAS use power-of-two maximum normalization. CEIL instead maps the exact maximum magnitude to 6, the largest value representable in E2M1, to avoid underutilizing the range.

Reducing the block size improves average QSNR under all three methods, but the gain depends on how the scale is selected. Under OCP, the average FTZ rate falls from 21.30% to 18.44%, yet QSNR improves by only 0.08 dB because saturation remains the dominant source of quantization

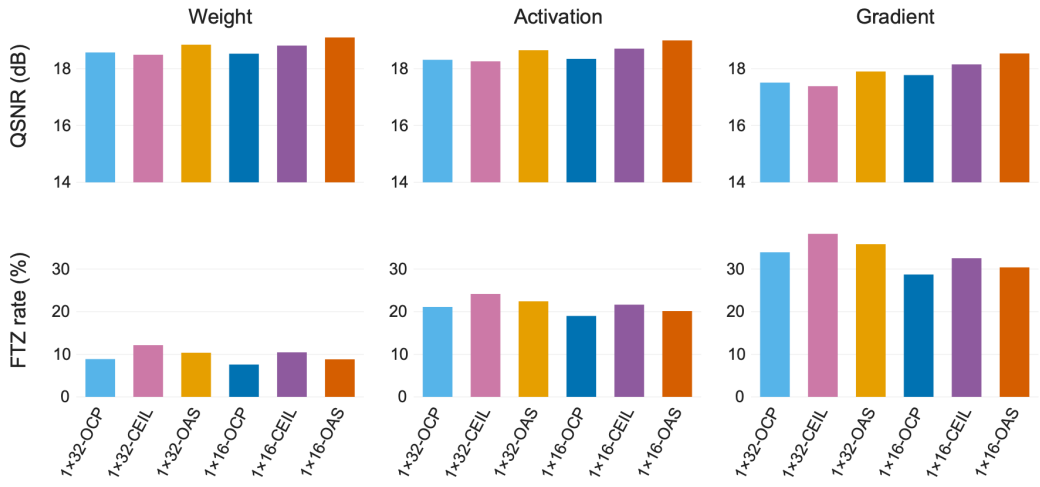


Figure 6: Tensor-level QSNR (top) and FTZ rate (bottom) for weights, activations, and gradients across MXFP4 configurations with different block sizes and scale-selection methods. Higher QSNR and lower FTZ rate are better.

error. Under CEIL, the same block-size reduction lowers FTZ from 24.85% to 21.54% and improves QSNR by 0.51 dB, substantially more than under OCP. This larger gain arises because the reduced dynamic range mitigates the loss of small values. Similar to CEIL, OAS benefits substantially from the smaller block, which lowers FTZ from 22.87% to 19.78% and improves QSNR by 0.41 dB.

Across both block sizes, OCP yields the lowest FTZ rate but remains limited by saturation, while CEIL eliminates saturation at the cost of the highest FTZ rate. By balancing these error sources, OAS achieves the highest average QSNR, exceeding OCP by 0.33 dB at 1×32 and 0.66 dB at 1×16. Combining OAS with 1×16 blocks therefore provides the best overall quantization quality.

D Runtime Analysis

We benchmark MXFP4 with MBS across macro-block sizes 64, 128, 256, and 512, using MXFP4, NVFP4, and BF16 GEMMs as references. All measurements are collected on a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU with 96 GB of memory using CUDA 13.2. Table 5 reports aggregate effective TFLOPS across 12 GEMM invocations for QKV projection, output projection, FC1, and FC2 in FPROP, DGRAD, and WGRAD for the 300M and 7B models. These invocations cover 11 distinct shapes for the 300M model and 12 for the 7B model. Runtime overhead is measured relative to MXFP4. MBS16 and MBS32 are omitted because supporting these sizes with the block-scaled FP4 MMA requires repeated MMA operations, which incur substantial overhead.

Across all macro-block sizes, the larger GEMMs in the 7B workload incur lower relative overhead than their 300M counterparts. For both workloads, the largest efficiency gain comes from increasing N from 64 to 128, which reduces overhead from 45.04% to 25.27% for 300M and from 42.42% to 23.18% for 7B. Beyond MBS128, the gains diminish, with MBS512 reducing the overhead only to 21.33% and 17.73%, respectively.

Projected end-to-end overhead. We apply FP4 quantization only to linear layers, so the measured GEMM overhead affects only this portion of a training step. In BF16 training, linear layers account for 26.36% of forward and backward time in the 300M model and 44.58% in the 7B model. We estimate the end-to-end impact by applying the measured MXFP4 and MBS128 GEMM performance to this portion while holding all other computation fixed. As shown in Table 6, the 25.27% and 23.18% GEMM overheads translate to only 2.61% and 4.51% end-to-end overhead for the 300M and 7B models, respectively.

Table 5: Effective TFLOPS and runtime overhead for the 300M and 7B model GEMM workloads. These kernel-level measurements do not represent end-to-end training overhead.

Format	300M		7B	
	TFLOPS	Overhead	TFLOPS	Overhead
BF16	367.54	+210.69%	388.57	+232.73%
NVFP4	1144.03	−0.19%	1327.61	−2.61%
MXFP4	1141.90	+0.00%	1292.91	+0.00%
MBS64	787.30	+45.04%	907.80	+42.42%
MBS128	911.53	+25.27%	1049.61	+23.18%
MBS256	912.68	+25.11%	1067.21	+21.15%
MBS512	941.18	+21.33%	1098.24	+17.73%

Table 6: Projected end-to-end overhead of MBS128 relative to MXFP4.

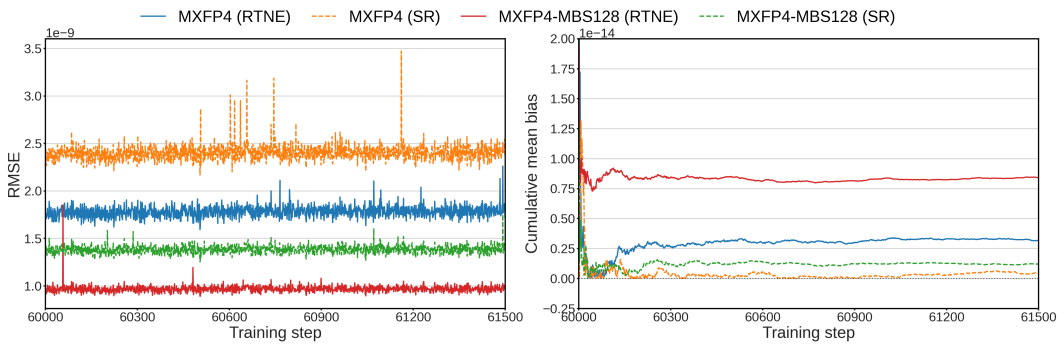
Model	Linear-layer share	Projected overhead
300M	26.36%	2.61%
7B	44.58%	4.51%

E Effect of Rounding Mode with MBS

To understand how rounding mode affects MBS, we examine the magnitude and bias of gradient quantization error, focusing on gradients in WGRAD. Let $\mathcal{E}_t$ denote the collection of element-wise errors across all FP4 linear layers at step t, where each error is the quantized gradient value minus its BF16 value. Let $\mathcal{E}_{\le t}$ denote the corresponding collection over all measured steps through step t. We compute the per-step RMSE and cumulative mean bias as

$$\mathrm{RMSE}_t = \sqrt{\frac{1}{|\mathcal{E}_t|}\sum_{e\in\mathcal{E}_t} e^2}, \qquad \mathrm{Bias}_t = \frac{1}{|\mathcal{E}_{\le t}|}\sum_{e\in\mathcal{E}_{\le t}} e. \tag{1}$$

RMSE measures the magnitude of quantization error at each step, whereas cumulative mean bias captures systematic error that persists across steps. Figure 7 shows that MBS128 lowers RMSE under both rounding modes. MXFP4-16-OAS-MBS128 has 45.65% lower RMSE under RTNE and 42.46% lower RMSE under SR than MXFP4-16-OAS. Under RTNE, however, its cumulative mean bias is 2.64× that of MXFP4-16-OAS at the final step. SR reduces the final bias by 86% for both methods. We attribute MBS128's higher RTNE loss in Figure 3(a) to its larger RTNE bias, and its greater benefit from SR to the larger absolute bias reduction.



(a) RMSE (b) Cumulative mean bias

Figure 7: We measure gradient quantization error for the four MXFP4 configurations in Figure 3(a) over 1,500 steps starting midway through training (steps 60,001–61,500). We report (a) RMSE at each training step and (b) cumulative mean bias over all measured errors through each training step.

F LONG-HORIZON EFFECT OF HADAMARD ROTATION

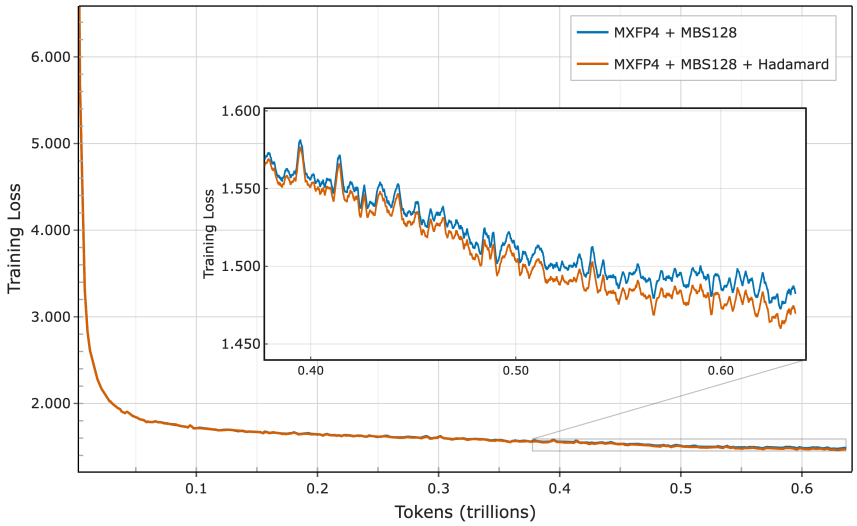


Figure 8: Training loss over 640B tokens for two matched 7B runs based on MXFP4-16-OAS-MBS128 and differing only in WGRAD Hadamard rotation. The inset enlarges the final stage of training.

Figure 3(b) selects WGRAD as the placement for Hadamard rotation, but its gain with MBS128 is small at 300M. The 7B ablation in Section 5.4 resumes from a checkpoint trained with WGRAD Hadamard rotation for 472B tokens. It therefore measures only the cost of disabling rotation late, rather than the value of applying it throughout training. To measure the latter, we compare matched 7B runs based on MXFP4-16-OAS-MBS128, with and without WGRAD Hadamard rotation over 640B tokens.

Figure 8 shows that the loss reduction from WGRAD Hadamard rotation increases steadily over training. The reduction grows from 0.04% over 100–150B tokens to 0.22% over 300–350B tokens and 0.82% over the final 50B tokens. This widening difference explains why removing Hadamard rotation in Section 5.4 produces only a modest loss increase. The 7B ablation branches resume from the same checkpoint at 472B tokens, after rotation has already been applied throughout the preceding training. The full trajectory therefore supports using WGRAD Hadamard rotation throughout FP4 training.