
NVFP4 Versus MXFP4 on Synthetic Weights: The Scale’s Precision Alone Closes Two Thirds of the Error Gap, the Smaller Block Alone a Quarter at Most

Yury Akinin (author) · Dmitrii Lozhkin (reviewer) · AVELIN AI · October 15, 2026

Abstract

MXFP4 and NVFP4 store a 4-bit value per weight and an 8-bit scale per block of weights: 32 weights and a power-of-two scale in MXFP4 (4.25 bits per weight), 16 weights and an FP8 scale in NVFP4 (4.5 bits). We vary the block size, the type of scale and the rule that picks the scale one at a time on seeded synthetic tensors, with a codec that matches one published NVFP4 shard byte for byte (maximum rule). On Gaussian data the relative weight error of MXFP4 and NVFP4 is 11.50% and 9.51% with the scale taken from the block maximum, and 11.16% and 8.12% with a least-error search. Giving MXFP4 only the FP8 scale, at its own block of 32 and the same number of bits, closes 66 to 69% of the gap on Gaussian and Student-t data (5 and 3 degrees of freedom). Giving it only the block of 16 closes 5.4% at most on Gaussian data and up to 24% on the Student-t data. The search is worth 14.6% to NVFP4 and 3.0% to MXFP4. On real matrices of two models (73,920 under the search, 1,728 also from the maximum; one model against its 8-bit weights) the mean NVFP4 error is within 0.04 percentage points of the Gaussian figures, and on one model the search cut output error by 13%. The ordering confirms earlier studies; the reference figures, the interaction and the check on real weights are what this note adds.

1 Introduction

Two block-scaled 4-bit formats are in use for running, and lately training, large language models. MXFP4, of the Microscaling family [1], gives 32 values one power-of-two scale. NVFP4 [2] gives 16 values one FP8 scale [3] and adds a scale for the whole tensor. NVFP4 is reported to be more accurate: on an 8B model MXFP4 needed 36% more training tokens to match its loss, in runs that also differed in another setting [4]. Which of the two matters more has been asked before. In training accuracy the format of the scale had a significant effect and the block size a modest one [5]. On tensors of two 8B models the mantissa of the scale was worth 3 to 4 dB and halving the block about 1 dB [6]. On the weights of one transformer block of one 8B model, with groups of 16 weights, a power-of-two scale added 40% to the squared error on average and an FP8 scale 10%, against an unquantized scale [7]. A search for the scale has been reported to cut the squared error of NVFP4 by 26 to 27% and of MXFP4 by 8% in simulation [8].

This note is a small confirmation of that picture on reference distributions, with three additions: a closed grid of block size, scale and rule on a codec that reproduces one published NVFP4 shard byte for byte (maximum rule); the interaction, to our knowledge not isolated before, that on Gaussian data the smaller block alone gains almost nothing under a power-of-two scale; and a check of the NVFP4 half against the real weights of two models, where the Gaussian figures hold to 0.04 pp and can serve, for these models, as constants for testing an encoder.

2 Setup

Formats. Every format here stores one E2M1 value per weight, a sign and one of the magnitudes 0, 0.5, 1, 1.5, 2, 3, 4 and 6, and one 8-bit scale per block of consecutive weights. The *block* is 32 weights (4.25 bits per weight) or 16 (4.5). The *scale* is a power of two, as in the MX formats [1]; an FP8 number with three mantissa bits (E4M3 [3]) under one FP32 scale per tensor, the tensor maximum divided by 2,688, as in NVFP4; or, as a limit, any real number. The *rule* either derives the scale from the largest magnitude in the block or searches for the scale of least squared error. For a power-of-two scale the first rule is the conversion of the MX specification [9]: the largest power of two not above the block maximum, divided by four, with values that land above 6 clamped; a third rule rounds that scale up so that nothing is clamped. For an FP8 scale the first rule is the block maximum divided by 6, rounded to the nearest FP8 value. The searches try 5 exponents, 17 FP8 codes or 73 real multipliers around the first rule’s choice; no block chose the edge of a window.

Data and metric. Synthetic tensors of $1,024 \times 4,096$ weights are drawn from a Gaussian and from Student-t distributions with 5 and 3 degrees of freedom, 5 seeds each. The metric is the relative Frobenius error of the decoded ten-

sor, $|W - \hat{W}| / |W|$. Across seeds no Gaussian figure moves by more than 0.011 percentage points (pp); at 3 degrees of freedom the spread reaches 0.20 pp. Figures derived from the tables use unrounded values. Real weights enter for NVFP4 only: the 192 MLP matrices of a 27B dense model [10] and the 73,728 expert matrices of a larger mixture-of-experts (MoE) model, one layer of which (1,536 matrices) was encoded under both rules.

Codec. Rounding to E2M1 and to FP8 comes from the codec that encoded the real weights. Under the maximum rule, and given the shard’s own tensor scales, it reproduces one shard of a publicly released NVFP4 checkpoint (384 matrices) byte for byte from its BF16 source; for the search rule there is no published file to match. An independent implementation that shares no code with the codec agrees with every synthetic figure to 0.001 pp on all 15 tensors.

3 Results

Table 1: Relative weight error (%) on Gaussian data. Bold: MXFP4 as specified (left) and NVFP4 (right).

Scale of a block	Block of 32 (4.25 bits per weight)		Block of 16 (4.5 bits per weight)	
	from maximum	by search	from maximum	by search
Power of two, as specified	11.50	11.16	11.78	10.99
Power of two, rounded up	11.54	–	11.28	–
FP8, 3 mantissa bits (E4M3)	10.16	9.09	9.51	8.12
Real number (limit)	10.11	8.89	9.41	7.85

The precision of the scale carries the gain. With the scale from the block maximum MXFP4 has an error of 11.50% and NVFP4 9.51%; with a search, 11.16% and 8.12% (Table 1, Figure 1a). Give MXFP4 only the FP8 scale, at its block of 32 and 4.25 bits per weight: the error falls to 10.16% and 9.09%, which closes 67% and 68% of the gap. Give it only the block of 16: 11.78% and 10.99%, which closes –14% and 5%. At a block of 16 the FP8 scale covers 96% and 91% of the distance from a power of two to a real-valued scale.

On Gaussian data a smaller block helps only under a precise scale. Under a power-of-two scale the rounded-up rule gains the most from the smaller block: 11.54% to 11.28%, or 2.2% for twice the scales. Under an FP8 scale the same step is worth 6.4% from the maximum and 10.6% with a search (9.09% to 8.12%). The last third of the gap between the formats is this step: the smaller block pays once the scale is fine.

The search pays where the scale is fine. Searching the scale cuts NVFP4 error by 14.6% and MXFP4 error by 3.0%: with only powers of two to choose from there is little to find. In squared error: 27% and 6%; [8] reports 26 to 27% and 8%, also in squared error, from simulation.

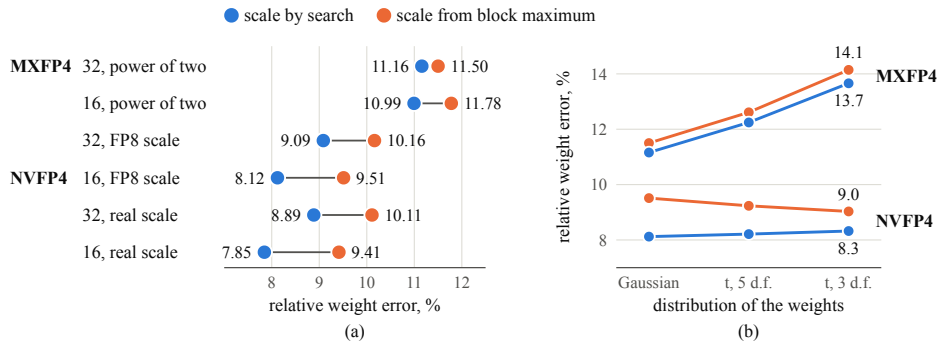


Figure 1: (a) Relative weight error on Gaussian data for each block size and type of scale, with the scale taken from the block maximum and found by search. (b) Both formats under both rules as the tails of the weight distribution grow heavier: Gaussian, then Student-t with 5 and 3 degrees of freedom (d.f.). Colours as in (a).

Real weights match the Gaussian figures. On the 27B model NVFP4 has a mean error of 9.49% from the maximum and 8.13% with a search; on a layer of the MoE model, against its 8-bit weights, 9.47% and 8.12%; over all 73,728 expert matrices 8.12% with a search, by the encoder’s own shard means, which lie between 8.12 and 8.13% (Table 2). The Gaussian figures are 9.51% and 8.12%, and no mean is further than 0.04 pp from them. The search won on each of the 1,536 matrices where both rules were run, by 1.0 to 1.4 pp. On the 27B model it also cut the mean output error of the 192 matrices on held-out activations from 7.86% to 6.82%, by 13% [10]. On the MoE layer it took 9.7 times the CPU time (1,225 against 126 s), one run each.

Under heavy tails the gap widens and the block begins to count. On synthetic Student-t data with 3 degrees of freedom MXFP4 rises to 14.1% and 13.7% while NVFP4 moves to 9.0% and 8.3% (Figure 1b): the formats are 1.57 and 1.64 times apart, against 1.21 and 1.37 on Gaussian data. The FP8 scale alone still closes 68% and 66% of the gap (69% and 68% at 5 degrees of freedom); the block of 16 alone now closes 15% and 24% (6% and 16%).

Table 2: NVFP4 relative weight error (%) on Gaussian data and on real weights (MoE model: against its 8-bit weights).

Weights	Matrices	Scale from maximum, %	Scale by search, %	Search gain, %
Gaussian noise, 5 tensors	5	9.51	8.12	14.6
27B dense model, all MLP matrices	192	9.49	8.13	14.4
MoE model, one layer of experts	1,536	9.47	8.12	14.3
MoE model, all expert matrices (encoder’s report)	73,728	–	8.12	–

4 Limitations

The comparison of formats is synthetic. MXFP4 was not run on real weights, and its figure for them is a prediction that rests on NVFP4’s agreement with the Gaussian case. The MoE figures are against 8-bit weights: against the original 16-bit weights a sample of 24 matrices of the layer gives 9.85% and 8.56%, the 2.7% error of the 8-bit source added in quadrature. The metric is weight error. It orders formats under the same kind of rule, but not encoders that use activations [10], and we give output error for one model and one pair of rules only. Model quality, activations and speed are not studied. The shares of the gap are for each change made alone, starting from MXFP4, and do not add up: starting from NVFP4, giving up the FP8 scale costs 76 to 114% of the gap and giving up the block of 16 costs 31 to 34%, over the three distributions and both rules; elsewhere the shares are not known. Earlier studies used other data and, in [6], leave the block size or scale format unstated, so comparisons with them are indicative.

5 Recommendations

1. Spend bits on the mantissa of the scale before the block size: an FP8 scale at a block of 32 beats a power-of-two scale at a block of 16 (9.09% against 10.99% on Gaussian data) with fewer bits.
2. With NVFP4, search the scale: 14.6% less weight error and, on the 27B model, 13% less output error, for about ten times the encoding CPU time (one MoE layer), paid once. With a power-of-two scale the gain is 3.0%.
3. Test an encoder two ways before a full build: reproduce one published shard byte for byte, and compare its decoded error with the constants (9.42 to 9.51% from the maximum and 8.12 to 8.14% with a search on every MLP matrix of the 27B model). A decoder that swaps the two codes of each byte reads the same file without complaint and returns 141%.

Disclosure. Experiments and analysis were produced with AI coding agents under the author’s direction. Numbers were recomputed from the raw records and from a seeded experiment that runs on a laptop CPU, except the errors of the MoE model, which are the encoder’s own report, checked against the written files on 24 matrices of one layer.

References

- [1] B. D. Rouhani, R. Zhao, A. More, et al. Microscaling data formats for deep learning. arXiv:2310.10537, 2023.
- [2] E. Alvarez, O. Almog, E. Chung, et al. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 24, 2025.
- [3] P. Micikevicius, D. Stosic, N. Burgess, et al. FP8 formats for deep learning. arXiv:2209.05433, 2022.
- [4] NVIDIA, F. Abecassis, A. Agrusa, et al. Pretraining large language models with NVFP4. arXiv:2509.25149, 2025.
- [5] B. Chmiel, M. Fishman, R. Banner, D. Soudry. FP4 all the way: Fully quantized training of LLMs. arXiv:2505.19115, 2025.
- [6] J. Chhugani, G. Jeong, B.-Y. Su, et al. Unveiling the potential of quantization with MXFP4: Strategies for quantization error reduction. arXiv:2603.08713, 2026.
- [7] V. Egiazarian, R. L. Castro, D. Kuznedelev, et al. Bridging the gap between promise and performance for microscaling FP4 quantization. In ICLR, 2026. arXiv:2509.23202.
- [8] T. Gupta, H. Prairie, X. Wu, et al. Search your block floating point scales! arXiv:2605.12464, 2026.
- [9] Open Compute Project. OCP Microscaling Formats (MX) Specification, Version 1.0. 2023.
- [10] Y. Akinin. Calibration leakage in a layer-wise quantization benchmark: With fewer calibration rows than input channels, in-sample scoring understates GPTQ error 2.6×. AVELIN AI Lab Research Note, October 2026.