



CONFERENCE

—NORTH AMERICA 2026—

Efficient Pretraining of LLMs in NVFP4

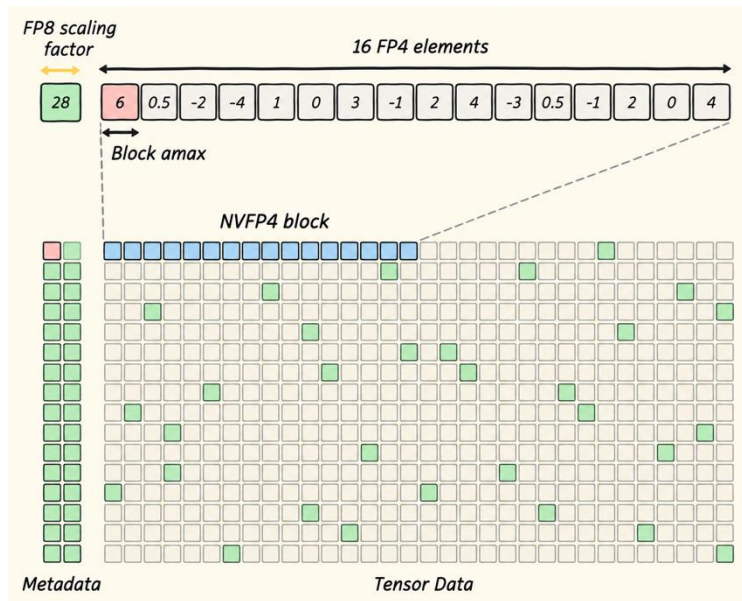
Anjulie Agrusa, Ryan Spring, Bruce Zitelli
NVIDIA

Agenda

- What is NVFP4 Pretraining?
- NVFP4 Pretraining Recipe
- Pretraining at Scale with NVFP4
- NVFP4 in the PyTorch ecosystem: TorchAO and TorchTitan
- Performance Analysis of PyTorch Implementation
- Reducing the Loss Gap to Higher Precision Training

What is NVFP4 Pretraining?

- **Format:** 4-bit E2M1 values, one E4M3 scale per 16 elements, and one FP32 scale per tensor
- **Method:** Quantize weights, activations, and gradients to NVFP4 to accelerate all 3 GEMMs: FPROP, WGRAD, DGRAD
- **Motivation:** GB300 offers 3× the peak dense throughput in FP4 versus FP8. Vera Rubin NVL72 raises peak dense FP4 throughput another 2.33× over GB300 NVL72.



A FP32 scale per tensor is applied (not pictured above)

[NVIDIA Blackwell](#), [NVIDIA Rubin](#)

[Pretraining Large Language Models with NVFP4, NVIDIA](#)

NVFP4 Pretraining Recipe

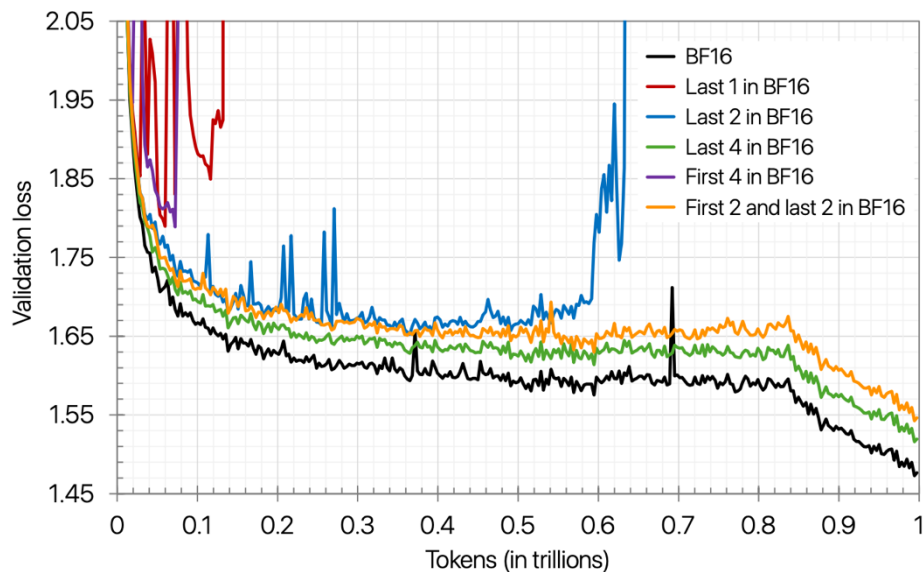
Stable low-precision training at scale

2D Weight Quantization 16x16 tiles	Random Hadamard Transforms on WGRAD GEMM	Stochastic Rounding on Gradient Tensors	High precision fallbacks
Consistent weight representations between forward and backward passes	Bounding block-level outliers	Reduce bias in the backward pass	Stability at scale

Released in TransformerEngine Sept. 2025

Pretraining Large Language Models with NVFP4, NVIDIA

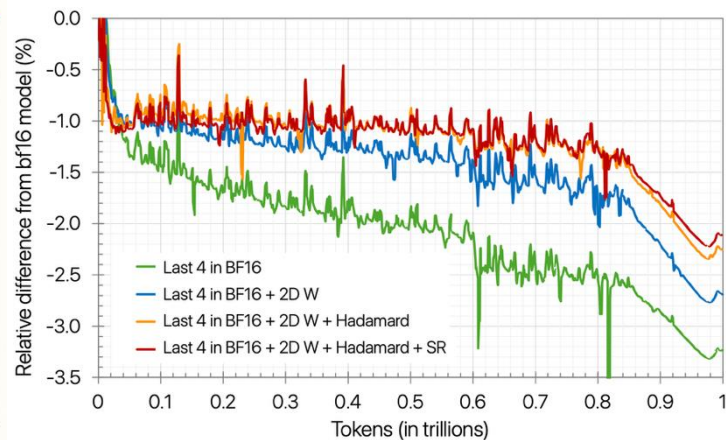
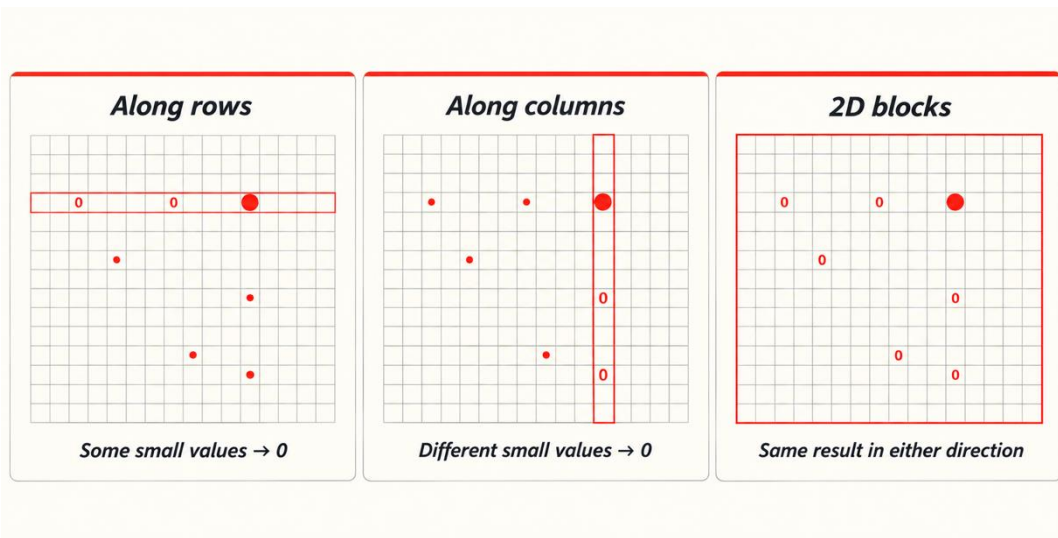
High Precision Fallbacks at End of the Network



Demonstrated on 1.2B dense transformer model

Pretraining Large Language Models with NVFP4, NVIDIA

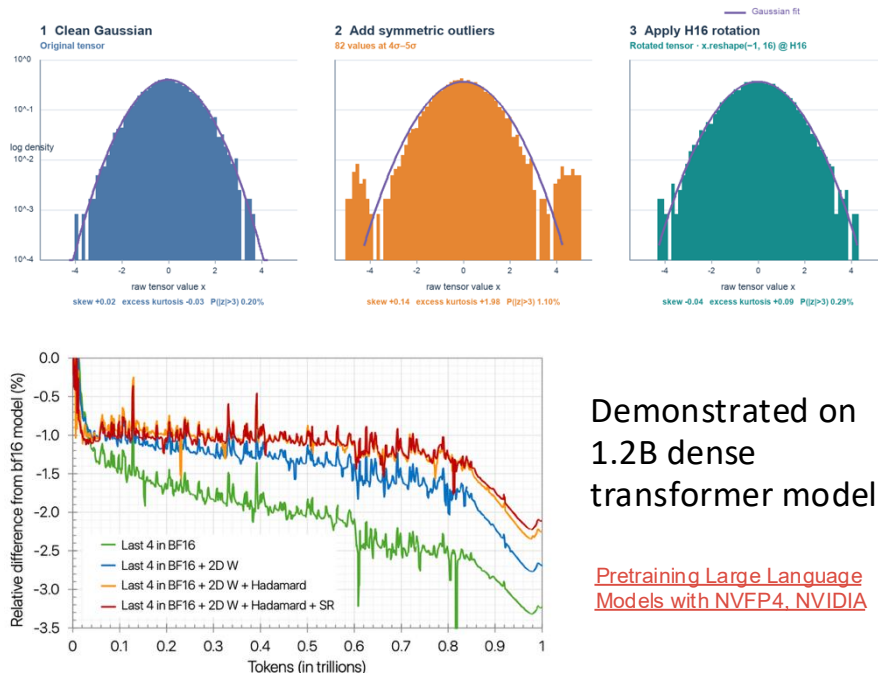
2D Weight Quantization in 16x16 Tiles



Demonstrated on 1.2B dense transformer model [Pretraining Large Language Models with NVFP4, NVIDIA](#)

Random Hadamard Transforms

- E2M1 format has limited dynamic range, compared to FP8 formats
- Mapping the block amax to the E2M1 max can cause small values to underflow
- Random Hadamard Transforms redistribute outlier energy to make distributions more Gaussian-like on aggregate, bounding block-level outliers



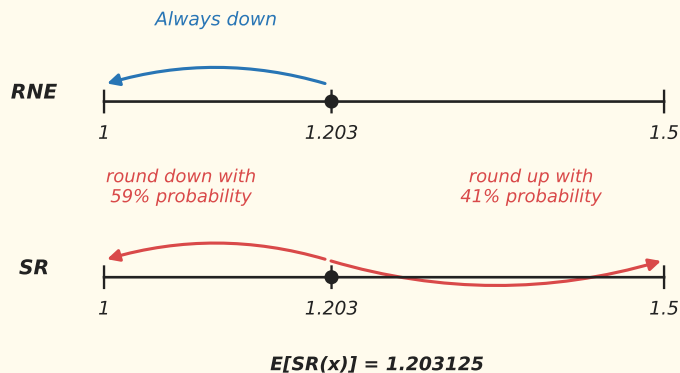
Demonstrated on
1.2B dense
transformer model

[Pretraining Large Language
Models with NVFP4, NVIDIA](#)

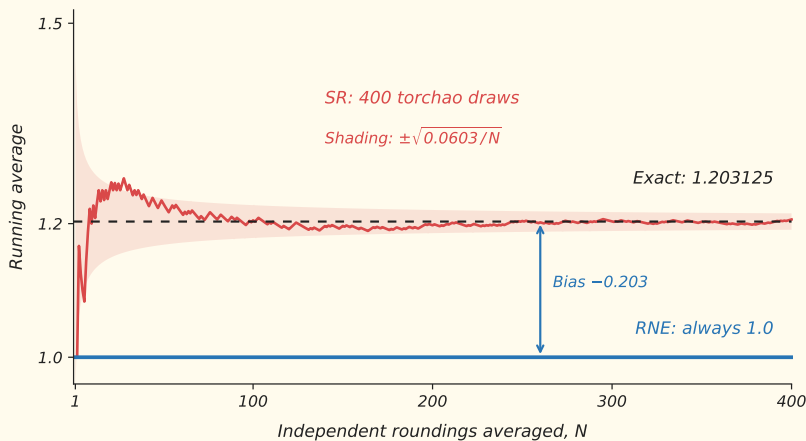
Stochastic Rounding

$FP4 \cdot scale = 1 \cdot x = 1.2$ (stored as 1.203 in BF16)

One rounding decision



After many trials, stochastic rounding recovers high precision value in expectation



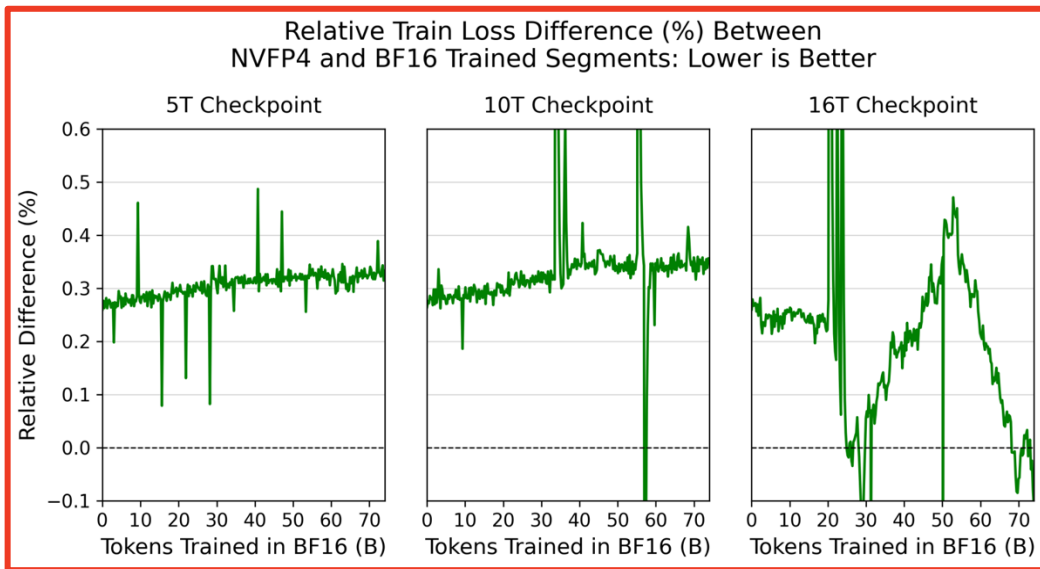
SR: bias ≈ 0 (0.003 after 400 draws) · RNE: bias -0.203

torchao NVFP4 quantizer · 400 SR draws · BF16 input

Pretraining at Large Model and Token Scale with NVFP4

Layer Type	Format	Rationale
All Linear Layers Unless Otherwise Noted	NVFP4	
Final 15% of Network	BF16	Promote training stability at scale
Latent Projections	BF16	Strategically kept in BF16 as step-time impact is negligible
MTP Layers	BF16	Preserves multi-token prediction capabilities
QKV & Attention Projections	BF16	Maintain fidelity of few attention layers
Mamba Output Projection	MXFP8	Mitigates high incidence of underflows observed when quantizing this layer to NVFP4 at smaller scales
Embedding Layers	BF16	

Nemotron 3 Super (120B-12B) and Ultra (550B-A55B) trained with NVFP4 recipe, 20-25T tokens



[Nemotron 3 Ultra, NVIDIA](#)
[Nemotron 3 Super, NVIDIA](#)

Where NVFP4 lives in PyTorch today

TorchTitan — model + distributed integration

- NVFP4 converter for Linear and MoE FFN layers

TorchAO — numerical recipe + operators

- NVFP4 recipe with CuTeDSL kernels

Designing Throughput Analysis for NVFP4 vs MXFP8

- Model: DeepSeek-V3 671B
- Hardware: GB300 with NVL interconnect
 - 64 nodes × 4 GPU/node = 256 GPUs
- EP64, TP1, PP1, Sequence length 4,096
- Vary microbatch size, larger number of tokens per GEMM
- 1 step warm up for torch.compile
- Throughput is measured every 10 steps as total work / total time

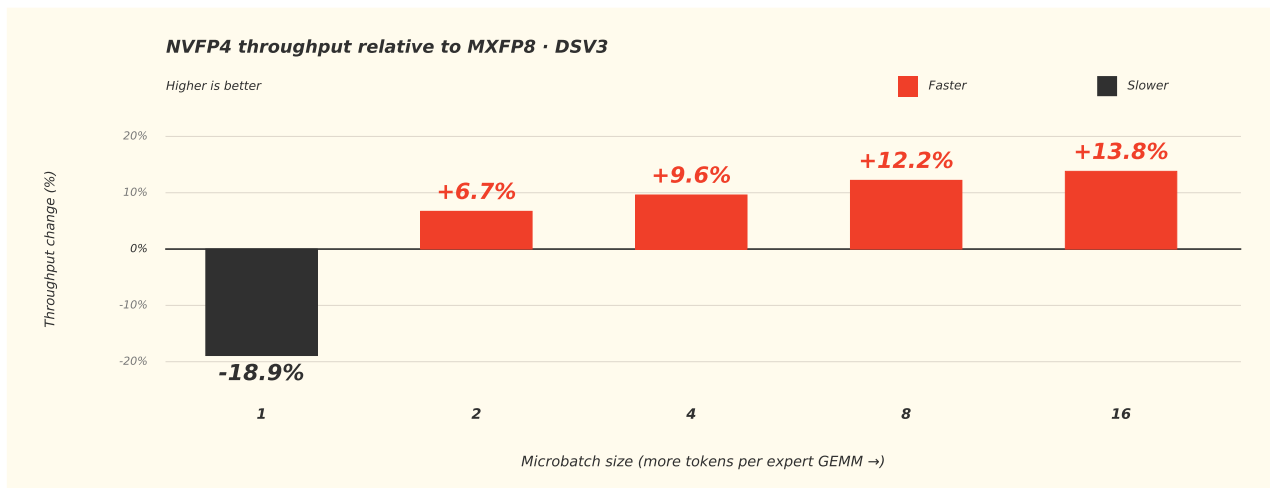
Quantization Recipes Used for Throughput Analysis

Operation	MXFP8 Recipe	NVFP4 Mixed Precision Recipe
FC1 GEMMs	MXFP8	NVFP4
FC2 GEMMs	MXFP8	NVFP4
FC3 GEMMs	MXFP8	NVFP4
Query Projection GEMMs	MXFP8	MXFP8
Key/Value Projection GEMMs	BF16	BF16
Attention Output Projection GEMMs	MXFP8	MXFP8
Attention BMMs	BF16	BF16

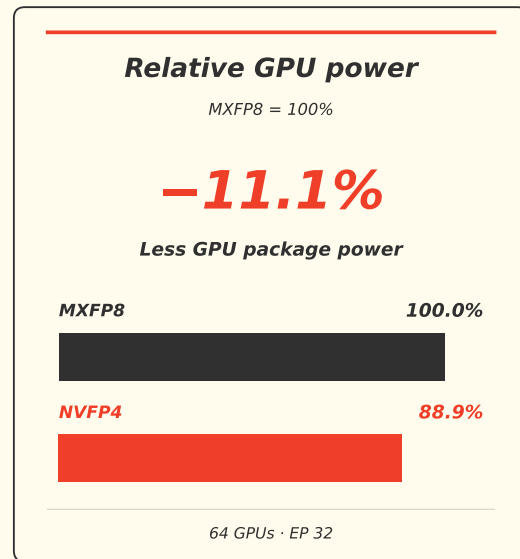
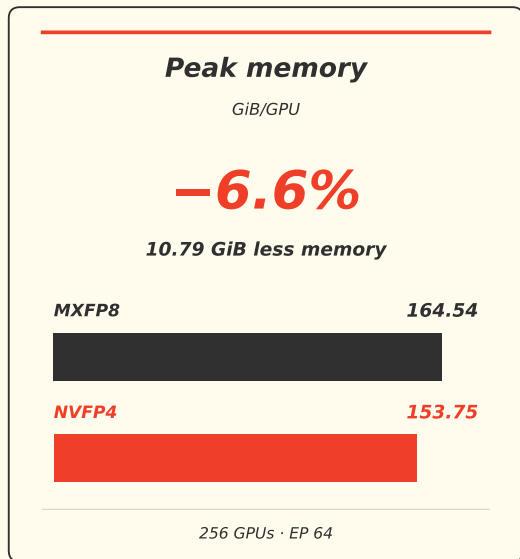
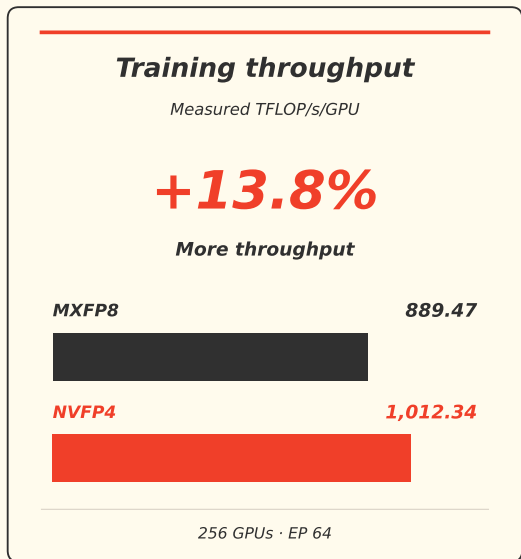
FC1/FC2/FC3 cover dense FFNs, shared experts, and routed experts

Measured Performance with PyTorch Implementation

- NVFP4 delivers increasing throughput gains at larger GEMM sizes
- MBS=16 maximizes improvement in throughput over tested sizes
- Limited in increasing MBS due to peak activation memory



A network with NVFP4 FFN layers improves over a MXFP8 network in 3 ways:



DeepSeek-V3 671B · GB300 · 256 GPUs: local batch 16 · M 131,072 | 64 GPUs: local batch 8 · M 32,768

Results are specific to the tested configuration and MoE backend.

Pretraining in NVFP4 needs a System of Optimizations



Numerics

*Carefully designed
training recipe*



Workload Optimization

*NVFP4 GEMM speedups observed
at large GEMM sizes*



Distributed Training Dynamics

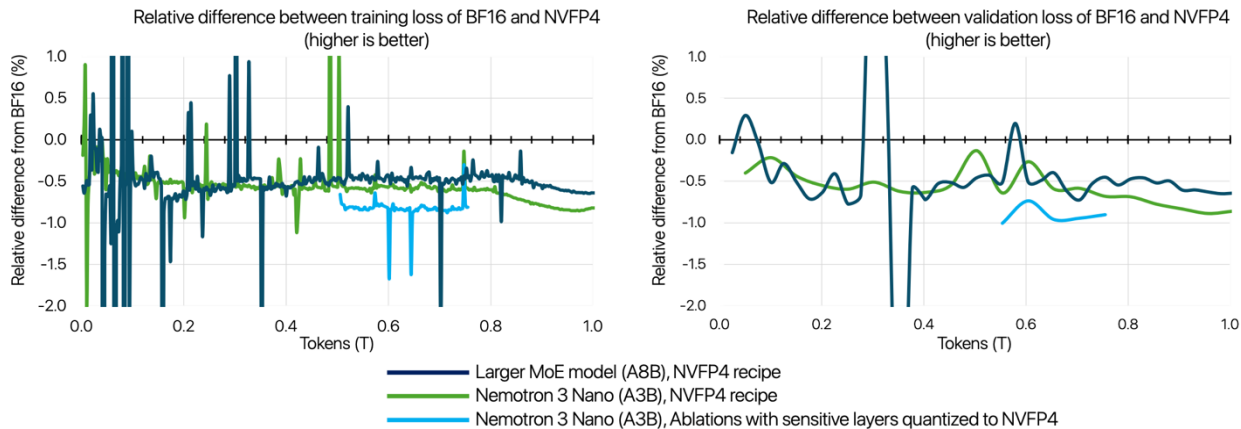
*Memory constraints and communication
bottlenecks impact how much of the GEMM
speedup reaches end-to-end training*



NVFP4-aware system design unlocks performance improvements

Motivation to Improve Loss Gaps Between NVFP4 and BF16

- NVFP4 production recipe demonstrated successful large-scale training
- However, a persistent loss gap remains relative to higher precision
- Smaller models may exhibit larger quantization-induced loss gaps



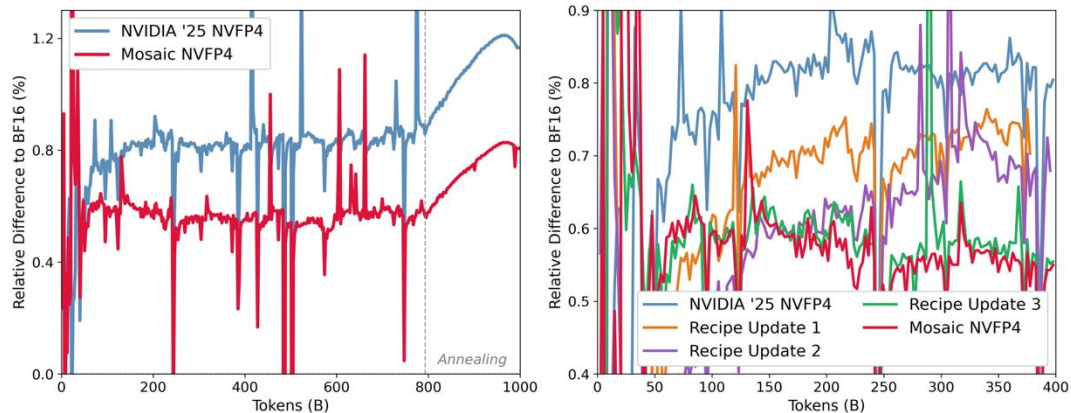
[NVIDIA Nemotron 3](#)

Reducing the Loss Gap to Higher Precision Training: NVFP4 Pretraining Research Community

- Quartet-II Backward Pass with MS-EDEN: *Strategy to address rounding bias; results in lower MSE than stochastic rounding*
 - (1) Random rotations on both backward GEMMs, (2) one-dimensional weight quantization with backward re-quantization
 - Training LLMs with MXFP4
 - Quartet
 - Tetrajet, Tetrajet v2
 - (3) Magnitude correction factor
- Four Over Six
 - Adaptive block-scale selection; improves MSE in select tensors

Reducing the Loss Gap to Higher Precision Training

30B-A3B Relative Train Loss Difference to BF16 (lower is better)



- All recipes have last 15% of the network in high precision
- Weights activations and gradients quantized to NVFP4 for all other linear layers in the transformer blocks

What Matters for NVFP4 Training?

Recipe	Weight Quant	Weight Re-Quant	Tiled Random Hadamard Transforms	Random Sign Vectors	Technique to Address Bias in Gradients	Four Over Six Tensors
NVIDIA '25 NVFP4	2D	No	Dim-16 WGRAD	Static	SR	None
Update 1	1D	Yes	Dim-16 WGRAD	Static	SR	None
Update 2	1D	Yes	Dim-128 WGRAD, DGRAD	Resampled	SR	None
Update 3	1D	Yes	Dim-128 WGRAD, DGRAD	Resampled	MS-EDEN	W.T, X.T
Mosaic NVFP4	1D	Yes	Dim-128 WGRAD, DGRAD	Resampled	MS-EDEN	W, X, W.T, X.T

Exciting roadmap for NVFP4
Pretraining ahead

Please visit us at the NVIDIA
booth!