Stable FP4 Training via Transposition-Invariant Block Quantization

Mehdi Rahimifar Amin Darabi Xing Huang Zhijun Tu Yunke Peng
Mehran Taghian Jazi Yao Wang Yufei Cui Hongliang Li

Abstract

Reducing training precision is a key lever for improving the efficiency of large language model (LLM) training, but pushing beyond FP8 to 4-bit floating point (FP4) remains challenging due to instability during optimization. We identify a fundamental source of this instability in existing microscaling approaches: *scale inconsistency induced by tensor transposition*. In conventional 1D block quantization, forward and backward passes assign different scaling factors to the same values after transposition, leading to biased and unstable gradient updates. To address this issue, we propose a low-precision training framework based on *2D block FP4 quantization*, which enforces transposition-invariant scaling and preserves consistency between forward and backward computations. We further combine this with truncation-free scaling and stochastic rounding to control quantization error and maintain unbiased gradients. To handle the sensitivity of attention mechanisms, we adopt MXFP8 quantization for query and key projections, yielding a practical mixed-precision design. We evaluate our method on dense LLMs up to 7B parameters and a 30B Mixture-of-Experts model, trained on up to 100B tokens. Across all settings, our approach achieves stable end-to-end FP4 training and closely matches BF16 performance, with less than 1.3% degradation in perplexity and downstream accuracy. These results demonstrate that enforcing forward–backward scaling consistency is sufficient to enable practical FP4 training at scale, providing a simple and effective pathway toward more efficient LLM training.

1 Introduction

Large Language Models (LLMs) have enabled major advances in natural language processing, including reasoning, code generation, and multimodal understanding [2, 7, 12]. However, these gains come at rapidly increasing computational and energy costs, making efficient training a central challenge in scaling LLMs [11, 10]. Reducing numerical precision has emerged as a key approach to improving efficiency, as lower-precision formats can significantly reduce memory bandwidth and increase arithmetic throughput [16, 9, 8]. While FP8 training is now practical on modern accelerators, pushing to 4-bit floating point (FP4) promises further reductions in memory footprint and compute [21, 13]. However, stable end-to-end FP4 training remains an open challenge due to limited dynamic range, large quantization error, and sensitivity during backpropagation and attention computation [4, 14].

Recent microscaling approaches, such as MXFP4 and NVFP4, address these challenges using block-wise quantization, where groups of values share a scaling factor [20, 17, 1]. While effective in reducing quantization error, these methods share a critical limitation: they rely on *one-dimensional (1D) block structures* that are not invariant to tensor transposition. During backpropagation, weight and activation matrices are transposed, reassigning values to different scaling blocks. This induces *scale inconsistency* between forward and backward passes, leading to biased gradients and degraded training stability as shown in Fig 1. **In this work, we identify transpose-induced scale**

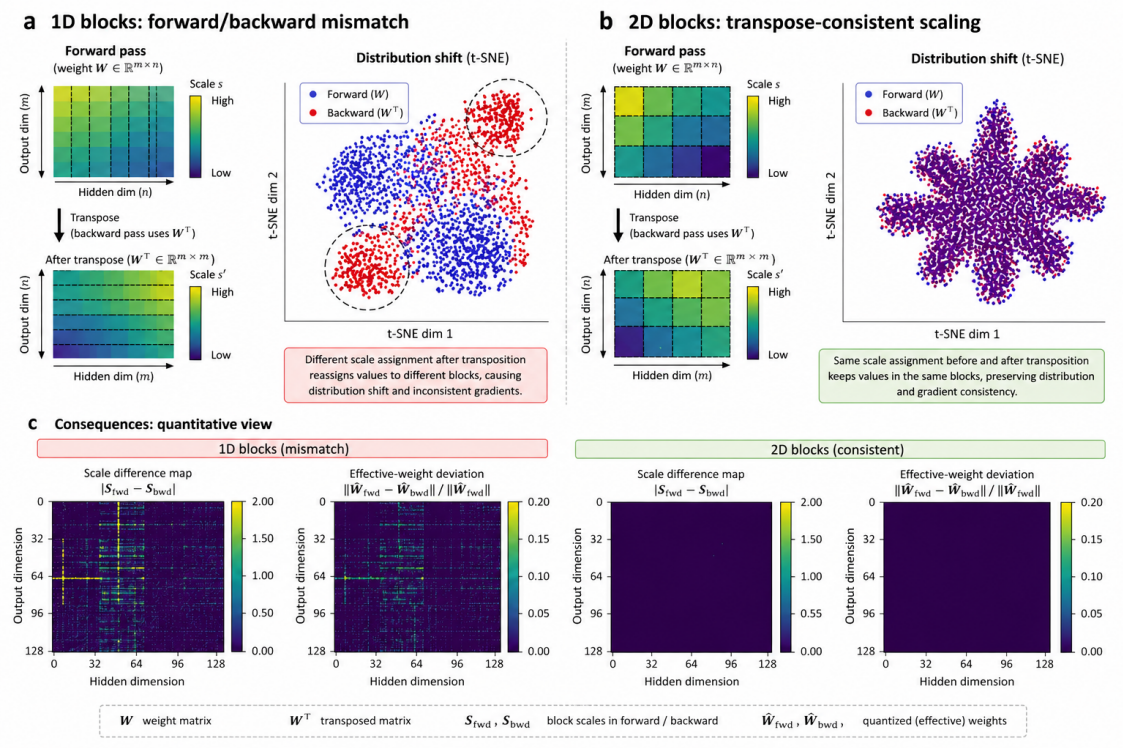


Figure 1: **Forward–backward inconsistency in 1D microscaling vs 2D blocks.** (a) In 1D block quantization, transposition reassigns values to different scaling blocks, causing forward–backward mismatch and distribution shift. (b) 2D block quantization preserves block structure under transposition, yielding consistent scaling and aligned distributions. (c) Scale difference maps show large mismatch in 1D but near-zero error in 2D. (d) Effective weight deviation is high for 1D blocks and negligible for 2D, confirming improved consistency.

inconsistency as a key failure mode in FP4 training. We show that 1D microscaling assigns inconsistent quantization scales across forward and backward passes, leading to biased gradients and unstable optimization. To address this, we propose a transposition-invariant FP4 training framework based on 2D block quantization, which enforces consistent scaling. We further combine this with truncation-free scaling and stochastic rounding, and adopt MXFP8 for query and key projections to stabilize attention. This design enables stable end-to-end FP4 training across models up to 30B parameters.

Contributions.

- **Scale inconsistency as a failure mode.** We identify transposition-induced scale mismatch in 1D microscaling as a key source of instability in FP4 training.
- **Transposition-invariant FP4 quantization.** We propose 2D block FP4 with truncation-free scaling and stochastic rounding to ensure consistent forward–backward behavior.
- **Practical mixed-precision FP4 training.** We combine FP4 linear layers with MXFP8 attention and demonstrate stable training up to 30B models with $\sim$1% gap to BF16.

2 Motivation for Microscaling

Efficient training of LLMs requires balancing numerical precision, dynamic range, and hardware efficiency. While reduced-precision formats such as FP16 and BF16 provide sufficient dynamic range, they remain costly in terms of memory bandwidth and compute, which increasingly dominate training at scale. Microscaling (MX) formats address this limitation by combining low-bit floating-point representations with block-wise scaling [15]. Instead of using a single global scale, MXFP assigns shared scaling factors to small groups of values, preserving local dynamic range while enabling aggressive precision reduction [13, 9]. Among these, MXFP4 is particularly attractive due to its extreme compression, using an `E2M1` representation with only four bits per value [18]. However, this aggressive quantization amplifies sensitivity to scaling errors, making stable training challenging.

Key limitation. Existing microscaling approaches primarily reduce quantization error *within* blocks, but largely overlook consistency *across* forward and backward passes. As we show below, this inconsistency becomes a dominant source of instability in FP4 training.

2.1 MXFP4

MXFP4 performs quantization using block-wise scaling:

$$P_i = \text{round}_{\text{FP4}}\left(\frac{X_i}{S}\right), \qquad X_i \approx P_i \cdot S,$$

where S is chosen based on the block maximum. NVFP4 improves representation fidelity by using smaller blocks and higher-precision scaling factors [1]. While this reduces intra-block quantization error, both approaches share a fundamental structural limitation.

Scale inconsistency under transposition. Both methods use one-dimensional (1D) block layouts (e.g., 1×32 or 1×16). During backpropagation, tensors are transposed, which reassigns values to different blocks and therefore different scaling factors. This induces a mismatch between forward and backward quantization:

$$\text{quant}_{\text{fwd}}(X) \neq \text{quant}_{\text{bwd}}(X),$$

leading to biased gradients and unstable optimization (Fig. 1). Reducing block size mitigates this effect but does not eliminate it [6], and introduces additional overhead. This reveals a fundamental requirement: *quantization schemes must be invariant to tensor transposition to ensure stable training.*

Bias induced by scale inconsistency. To understand this instability, consider a tensor X quantized in the forward pass with scale S_{fwd} and in the backward pass with a different scale S_{bwd}. The resulting quantized values are

$$\hat{X}_{\text{fwd}} = \text{round}\left(\frac{X}{S_{\text{fwd}}}\right) S_{\text{fwd}}, \quad \hat{X}_{\text{bwd}} = \text{round}\left(\frac{X}{S_{\text{bwd}}}\right) S_{\text{bwd}}.$$

When $S_{\text{fwd}} \neq S_{\text{bwd}}$, the forward and backward passes operate on different quantized representations. As a result, the backward pass effectively computes gradients with respect to a *different quantized objective* than the one used in the forward pass.

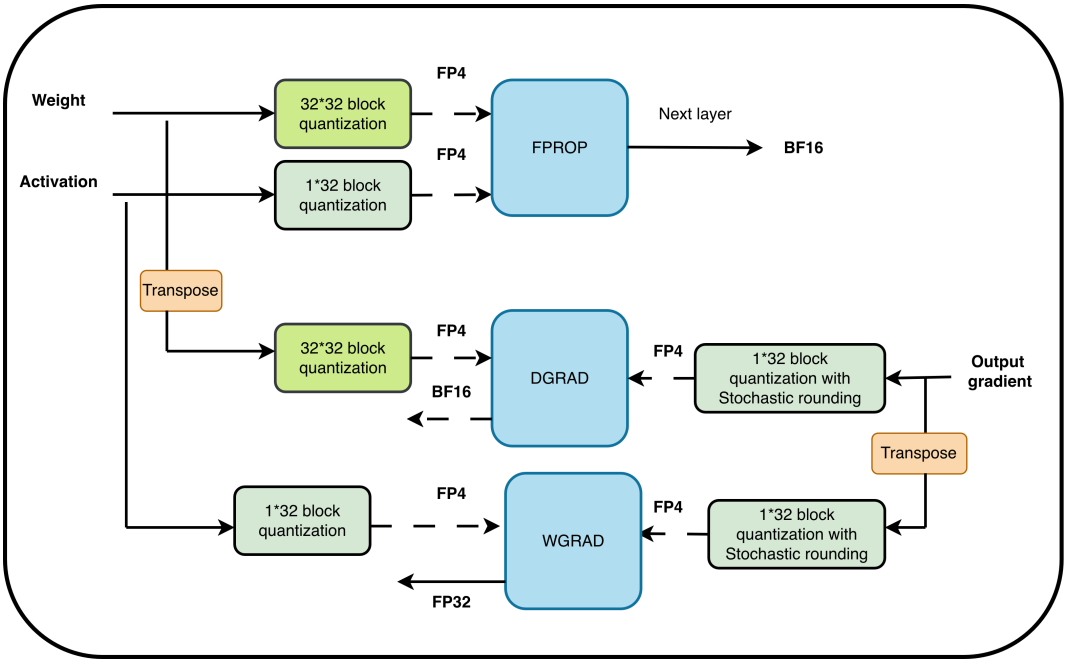


Figure 2: **Overview of 2D block FP4 quantization.** Tensors are partitioned into square blocks that share a scale, enabling consistent scaling across forward and backward passes.

Under the straight-through estimator (STE) [3], gradients are propagated as if $\hat{X} \approx X$, but this mismatch introduces a systematic bias:

$$\nabla \mathcal{L}_{\text{bwd}} \neq \nabla \mathcal{L}_{\text{fwd}}.$$

In contrast, if the quantization scheme is invariant to transposition (i.e., $S_{\text{fwd}} = S_{\text{bwd}}$), both passes operate on a consistent representation, eliminating this source of bias.

Implication. These observations suggest that stable FP4 training requires not only low quantization error, but also *forward–backward consistency*. This motivates the design of transposition-invariant quantization schemes, which we achieve using 2D block structures in the next section.

3 Transposition-Invariant Block Quantization

We propose a transposition-invariant quantization scheme based on *2D square blocks*. Unlike 1D layouts, square blocks preserve their structure under transposition, ensuring that values are associated with the same scaling factors in both forward and backward passes. The overall flow is illustrated in Fig 2

Transposition invariance. Let $X \in \mathbb{R}^{m \times n}$ be partitioned into square blocks of size $b \times b$. A block $B_{i,j}$ in X maps under transposition to the block $B_{j,i}$ in $X^\top$. Because $B_{j,i}$ contains the same values as $B_{i,j}$ up to transposition, the block maximum and therefore the block scale are preserved:

$$S(B_{i,j}) = S(B_{i,j}^\top).$$

This preserves the scale assignment of values across forward and backward computations. In contrast, 1D layouts generally regroup values after transposition, producing inconsistent quantization ranges.

Application to linear layers. For a linear layer:

$$Y = XW^\top, \quad \nabla X = \nabla Y W, \quad \nabla W = (\nabla Y)^\top X,$$

we apply 2D FP4 quantization to weights and gradients. We use 32×32 blocks for weights to align with hardware tiling. For gradients, we apply 2D quantization to both weight gradients and data gradients, enabling low-precision computation throughout the backward pass. We retain a 1D layout for activations to avoid mixing semantically unrelated values across channels. This reflects a trade-off between statistical structure and quantization consistency. To enable end-to-end training, we use the Straight-Through Estimator (STE) [3]. Since quantization is piecewise constant and non-differentiable, directly computing gradients would block optimization. The STE approximates the derivative of the quantization function as identity during backpropagation. Concretely, for $\hat{X} = \text{quant}(X)$, we use

$$\frac{\partial \mathcal{L}}{\partial X} \approx \frac{\partial \mathcal{L}}{\partial \hat{X}}.$$

This allows gradients to pass through the quantization step while retaining low-precision computation in the forward pass.

3.1 Truncation-Free Scaling and Rounding

The stability of low-precision training depends not only on block structure, but also on how values are scaled and rounded during quantization. In FP4, the limited dynamic range makes the system particularly sensitive to overflow and rounding bias, both of which can destabilize optimization if not properly controlled. To address overflow, we adopt truncation-free scaling. Specifically, we compute the scale as

$$S = 2^{\lceil \log_2(2M/(Q_p - Q_n)) \rceil},$$

where M is the maximum absolute value within a block and (Q_p, Q_n) denote the positive and negative bounds of the FP4 format [5]. This choice ensures that all values fall within the representable range, preventing clipping and avoiding systematic distortion of large-magnitude activations and gradients. To mitigate bias introduced by quantization, we use stochastic rounding during backpropagation [6]. Instead of deterministically mapping values to the nearest representable level, stochastic rounding preserves expectations:

$$\mathbb{E}[\text{roundS}(x)] = x.$$

This property is critical for maintaining unbiased gradient estimates, especially under aggressive quantization. Taken together, truncation-free scaling and stochastic rounding provide complementary benefits. The former eliminates instability caused by overflow, while the latter reduces bias in gradient estimation. In combination with transposition-invariant quantization, these mechanisms enable stable optimization under FP4 precision.

3.2 MXFP8 Attention

Attention computation is particularly sensitive to quantization due to two compounding effects: the dot-product interaction $QK^\top$ amplifies quantization noise, and the softmax normalization further magnifies small perturbations. As a result, aggressively reducing precision in this pathway often leads to instability or degraded model quality [19]. In practice, we observe that directly quantizing query (Q) and key (K) projections to FP4 introduces errors that propagate through $QK^\top$ and distort the attention distribution.

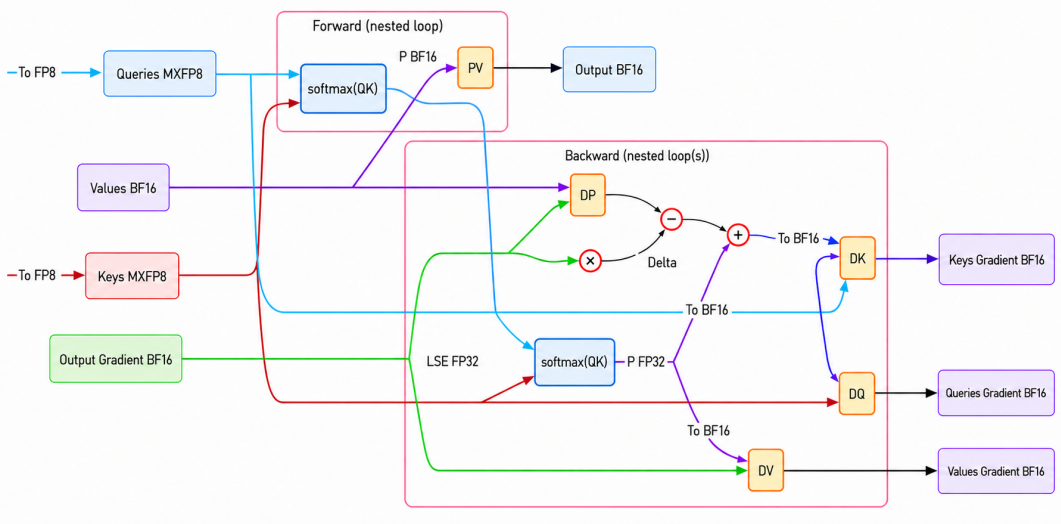


Figure 3: **MXFP8 attention pathway.** Query and key projections are quantized using MXFP8 to improve numerical stability while retaining low-precision computation in the rest of the model.

Table 1: Theoretical efficiency comparison on OLMo-1B. Our method applies MXFP8 to Q/K linear layers and 2D-FP4 to all other transformer linear layers. Memory values reflect parameter storage only; throughput estimates are idealized.

Metric	BF16	Our method	Saving / speedup	Notes
Q/K linear weights	268.4 MB	134.2 MB	50.0% smaller	MXFP8
Other attention linear weights	268.4 MB	67.1 MB	75.0% smaller	2D-FP4
MLP linear weights	1610.6 MB	402.7 MB	75.0% smaller	2D-FP4
Transformer linear weights	2147.4 MB	604.0 MB	71.9% smaller	mixed
Total model weights	2353.5 MB	810.0 MB	65.6% smaller	embeddings BF16
Linear activation bandwidth	1.00×	$\sim 0.30\times$	$\sim 70\%$ lower	weighted avg.
Ideal linear throughput	1.00×	$\sim 3.56\times$	$\sim 3.56\times$ faster	ideal kernels

While our 2D block FP4 quantization resolves forward–backward scale inconsistency in linear layers, it does not fully address this sensitivity in attention. To mitigate this issue, we adopt a mixed-precision design in which Q and K are quantized using MXFP8, while all other transformer linear layers (including MLP and attention value/output projections) use 2D block FP4. This preserves higher precision where it is most critical, while retaining the efficiency benefits of FP4 for the dominant matrix multiplications. The overall attention pipeline is illustrated in Figure 3. Despite using higher precision for Q and K, the overhead of this hybrid design remains modest. Empirically, we observe that the total training memory remains close to the FP4-only configuration, while providing improved numerical stability.

To quantify the efficiency implications of this design, Table 1 presents a theoretical comparison against BF16 training on OLMo-1B. We report weight memory and idealized throughput for transformer linear layers, which dominate training cost. Because the majority of parameters reside in MLP layers, which are fully quantized to FP4, the overall system retains most of the benefits of aggressive low-precision training. In this setting, our method achieves an estimated $\sim$65% reduction in model weight memory and up to $\sim 3.6\times$ idealized throughput improvement for linear operations, while only requiring FP8 precision in a small fraction ($\sim$12.5%) of parameters.

Overall, combining 2D block FP4 quantization with MXFP8 attention yields a system that targets the two dominant costs in transformer training: matrix multiplications ($\mathcal{O}(Nd^2)$), which benefit from FP4, and attention computation ($\mathcal{O}(N^2d)$), which benefits from higher precision in $QK^\top$. This balance enables stable end-to-end FP4 training while maintaining a favorable trade-off between efficiency and model quality.

4 Experiments

We evaluate whether enforcing forward–backward scaling consistency enables stable end-to-end FP4 training across model scales and architectures. Our experiments are designed to answer four questions: (1) Does 2D block FP4 eliminate the instability observed in conventional FP4 training? (2) How closely does the proposed method match BF16 training in convergence and final quality? (3) What is the contribution of truncation-free scaling, stochastic rounding, and MXFP8 attention? (4) How does the method behave across dense and mixture-of-experts models?

4.1 Experimental Setup

We consider three representative settings spanning both dense and sparse transformer architectures: OLMo-1B, OLMo-7B, and Qwen 30B MoE. These experiments allow us to assess whether the proposed method remains stable as model size and architectural complexity increase.

For each model, we compare the following configurations:

1. **BF16**: a full-precision training baseline.
2. **2D-FP4**: our proposed 2D block FP4 quantization applied to linear layers in both forward and backward passes.
3. **2D-FP4 + MXFP8**: the same configuration augmented with MXFP8 quantization for attention queries and keys.

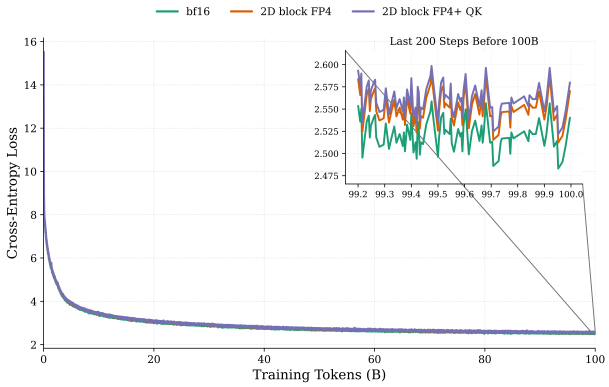
For OLMo-1B, we train from scratch on a 100B-token corpus. We use the same token budget for OLMo-7B to evaluate scaling behavior in larger dense models, and for Qwen 30B MoE to assess robustness in a sparse mixture-of-experts setting.

Because native FP4 tensor-core support is not available on our training platform, we emulate microscaling behavior in software while training. Consequently, our experiments focus on *optimization stability* and *model quality*, rather than realized hardware speedup. Our method applies 2D block FP4 quantization consistently across forward and backward passes for linear layers. In the hybrid configuration, attention queries and keys are quantized using MXFP8, allowing us to isolate the benefit of higher precision in the most numerically sensitive part of the transformer. We evaluate models on both language modeling and downstream reasoning tasks. Language modeling performance is measured using perplexity on Wikitext, Pile, and C4, reported in Table 2. Downstream reasoning performance is measured using accuracy on SciQ, COPA, ARC-Easy, and HellaSwag, reported in Table 3. Training dynamics across model scales are shown in Figure 4, and ablations on quantization design choices are shown in Figure 5.

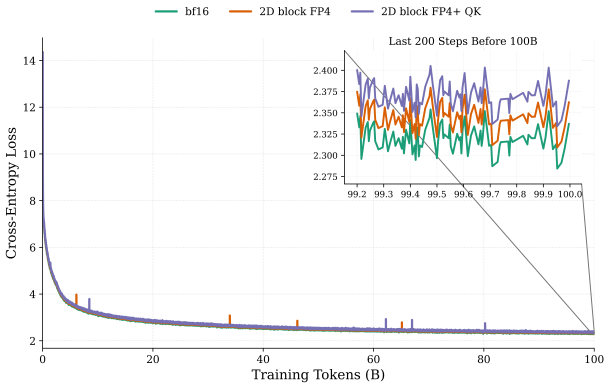
4.2 Main Results

We evaluate the proposed method along four dimensions: training stability, the effect of quantization design choices, language modeling performance, and downstream reasoning ability. Across all

(a) OLMo-1B



(b) OLMo-7B



(c) Qwen 30B MoE

Figure 4: **Training dynamics under FP4 quantization across model scales.** Naive FP4 without microscaling diverges early, while microscaling-based methods remain stable. The proposed 2D block FP4 method consistently converges closest to BF16, with the performance gap decreasing as model size increases.

experiments, we compare BF16 training with our 2D block FP4 method, with and without MXFP8 attention.

We begin by analyzing training stability and convergence. Figure 4 shows the training dynamics across all model scales. In the OLMo-1B setting (Figure 4a), naive FP4 quantization without microscaling diverges early, with instability appearing at approximately 20B tokens. In contrast, all microscaling-based methods remain stable throughout training, confirming that block-wise scaling is necessary for FP4 optimization. Among these stable approaches, our 2D block FP4 method consistently tracks the BF16 baseline more closely than alternative microscaling schemes. At 1B scale, the final loss gap is approximately 1.1%, compared to around 2% for competing methods. Incorporating MXFP8 attention slightly increases the gap to 1.6%, but remains substantially closer to BF16 than prior FP4 baselines. This suggests that enforcing transposition-invariant scaling is the primary factor driving stability, while higher-precision attention provides an additional but secondary benefit. This behavior remains consistent at larger scales. As shown in Figures 4b and 4c, both OLMo-7B and Qwen 30B MoE converge stably under FP4 training, with final loss gaps of approximately 1% relative to BF16. Notably, the gap decreases as model size increases, suggesting that larger models are more tolerant to quantization noise once scaling consistency is enforced.

We next examine the contribution of truncation-free scaling and stochastic rounding. Figure 5

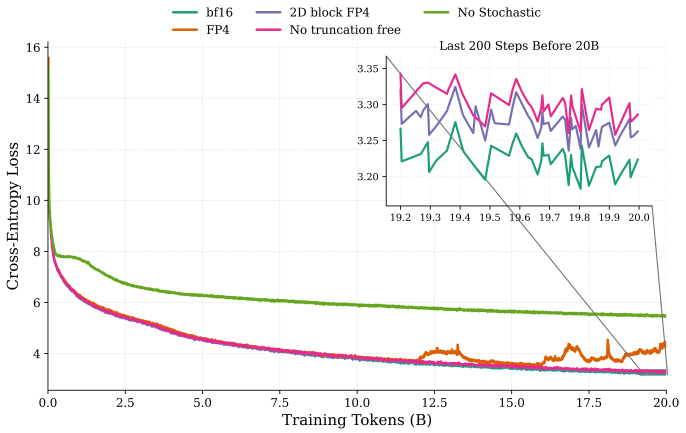


Figure 5: **Ablation of truncation-free scaling and stochastic rounding.** Both components are necessary for stable FP4 training: removing truncation-free scaling leads to overflow-induced instability, while disabling stochastic rounding introduces bias and degrades convergence.

Table 2: **Language modeling results** (perplexity; lower is better).

Model	Method	Δ	Wikitext	Pile	C4	Avg
OLMo-1B	BF16	–	18.66	11.72	18.19	16.19
	Ours	1.1	18.82	11.80	18.57	16.40
	+MXFP8	1.6	19.13	11.94	18.71	16.59
OLMo-7B	BF16	–	16.22	10.60	15.92	14.25
	Ours	0.9	16.30	10.66	16.05	14.34
	+MXFP8	1.2	16.45	10.74	16.18	14.46
Qwen 30B	BF16	–	15.08	9.65	14.96	13.23
	Ours	0.8	15.15	9.70	15.05	13.30
	+MXFP8	1.1	15.28	9.78	15.18	13.41

shows that removing truncation-free scaling leads to instability due to numerical overflow, while disabling stochastic rounding introduces bias that degrades convergence. When both components are used together, the model achieves stable optimization and the smallest loss gap relative to BF16. These results highlight that stable FP4 training requires not only consistent scaling across forward and backward passes, but also careful control of overflow and rounding bias.

We then evaluate final model quality on language modeling benchmarks. Table 2 reports perplexity on Wikitext, Pile, and C4. Across all model scales, our method closely matches the BF16 baseline. For OLMo-1B, the average degradation is approximately 1.1%, while for OLMo-7B and Qwen 30B the gap decreases to below 1%. The hybrid 2D-FP4 + MXFP8 configuration exhibits slightly higher perplexity than the base 2D-FP4 variant, but remains within a narrow margin of BF16 across all datasets. Overall, these results show that aggressive precision reduction does not significantly degrade language modeling performance when scaling consistency is enforced.

We further evaluate downstream reasoning performance. Table 3 reports accuracy on SciQ, COPA, ARC-Easy, and HellaSwag. Across all tasks, our method remains close to BF16, with performance differences generally within approximately 1 percentage point. In several cases, particularly on SciQ and COPA, the FP4 model slightly exceeds the BF16 baseline. While modest, this improvement may be attributed to a regularization effect induced by quantization noise. The MXFP8 variant exhibits slightly lower accuracy than the base FP4 configuration, but remains com-

Table 3: **Reasoning benchmark results** (accuracy; higher is better).

Model	Method	SciQ	COPA	ARC-E	HellaSwag	Avg
OLMo-1B	BF16	77.60	70.00	51.75	45.52	61.22
	Ours	78.21	72.00	50.00	44.62	61.21
	+MXFP8	77.22	69.00	49.64	44.15	60.00
OLMo-7B	BF16	79.20	72.00	54.21	50.86	64.07
	Ours	79.80	73.00	53.80	50.30	64.23
	+MXFP8	79.10	71.00	53.20	49.70	63.25
Qwen 30B	BF16	85.40	79.31	60.79	56.96	70.62
	Ours	85.90	80.10	60.40	56.50	70.73
	+MXFP8	85.20	78.90	59.80	55.90	69.95

Table 4: **Block-size trade-offs in 2D FP4 quantization.**

Block size	Final loss gap (%)	Relative memory	Relative throughput
8×8	∼0.8–1.0	∼0.30×	∼0.85×
16×16	∼0.9–1.1	∼0.27×	∼0.92×
32×32	1.1	∼0.25×	1.00×
64×64	∼1.3–1.6	∼0.24×	∼1.05×

petitive overall. We also analyze the trade-off between block size, accuracy, and efficiency. Table 4 summarizes trends for different 2D block sizes. Smaller blocks improve numerical fidelity but increase metadata overhead and reduce throughput, while larger blocks improve efficiency at the cost of increased loss gap. The 32×32 configuration used in our experiments provides a practical balance between stability and efficiency. Taken together, these results demonstrate that enforcing forward–backward consistent scaling enables stable end-to-end FP4 training while preserving model quality across both dense and MoE architectures. Once scale inconsistency is removed, FP4 training can closely approach BF16 performance.

5 Discussion and Limitations

Our results suggest that the main challenge in FP4 training is not simply limited numerical precision, but a structural mismatch between quantization schemes and backpropagation. By enforcing transposition-invariant scaling through 2D block quantization, we eliminate a key source of gradient bias and enable stable optimization at 4-bit precision. This perspective reframes low-precision training as a problem of *consistency*, rather than purely representational fidelity. The proposed approach is simple and broadly applicable, requiring minimal modifications to existing training pipelines while delivering consistent improvements across model scales and architectures. Notably, the performance gap between FP4 and higher-precision training decreases as model size increases, suggesting that larger models are inherently more robust to quantization noise once systematic bias is removed. Furthermore, our mixed-precision treatment of attention demonstrates that different components of the transformer exhibit varying sensitivity to aggressive quantization, indicating that selective precision allocation is an effective and practical strategy.

Despite these encouraging findings, several limitations remain. First, our experiments rely on software emulation of FP4 behavior on hardware that does not natively support the MXFP4 format. Real-world efficiency gains will require next-generation accelerators with native support, where microscaling formats can be executed efficiently to fully realize the speed and energy benefits of

FP4 training. Second, while 2D block quantization improves consistency in matrix operations, it introduces additional design trade-offs, including the choice of block size and the overhead of storing scaling factors. The optimal configuration is likely to depend on the specific architecture and workload, and may require further tuning in practice. Finally, our evaluation is limited to models up to 30B parameters and training runs of up to 100B tokens. Scaling to larger, frontier-scale models and longer training regimes may expose additional stability challenges. Future work should explore adaptive or learned block structures, closer hardware–software co-design, and the possibility of further reducing precision in sensitive components such as attention.

6 Conclusion

We presented a low-precision training framework for large language models based on transposition-invariant 2D block FP4 quantization, combined with MXFP8 for attention. Our central insight is that instability in existing FP4 methods arises not only from limited precision, but from scale inconsistency between forward and backward passes induced by tensor transposition. By replacing 1D microscaling with 2D square blocks, our method enforces consistent scaling and substantially improves optimization stability. Across dense models up to 7B parameters and a 30B Mixture-of-Experts model trained on up to 100B tokens, our approach enables stable end-to-end FP4 training while closely matching BF16 performance, with less than 1.3% degradation. Ablations further show that truncation-free scaling and stochastic rounding are important complementary components for achieving robust convergence under extreme quantization. More broadly, our results suggest that the main obstacle to practical FP4 training is not simply quantization error, but inconsistency in scaling across forward and backward computations. This perspective shifts the design of low-precision training from improving local representational fidelity to enforcing global computational consistency. We believe this view opens promising directions for future work, including hardware support for transposition-consistent quantization and further scaling to larger models.

References

[1] 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. *arXiv preprint arXiv:2509.25149*, 2025.

[2] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Roberto L Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native fp4 training can be optimal for large language models. *arXiv preprint arXiv:2505.14669*, 2025.

[5] Yuxiang Chen, Haocheng Xi, Jun Zhu, and Jianfei Chen. Oscillation-reduced mxfp4 training for vision transformers. *arXiv preprint arXiv:2502.20853*, 2025.

[6] Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. Fp4 all the way: Fully quantized training of llms. *arXiv preprint arXiv:2505.19115*, 2025.

[7] Ruihao Gong, Yifu Ding, Zining Wang, Chengtao Lv, Xingyu Zheng, Jinyang Du, Yang Yong, Shiqiao Gu, Haotong Qin, Jinyang Guo, et al. A survey of low-bit large language models: Basics, systems, and algorithms. *Neural Networks*, page 107856, 2025.

[8] Zhiwei Hao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, Guoxia Wang, Dianhai Yu, Yonggang Wen, and Dacheng Tao. Low-precision training of large language models: Methods, challenges, and opportunities. *arXiv preprint arXiv:2505.01043*, 2025.

[9] Alejandro Hernández-Cano, Dhia Garbaya, Imanol Schlag, and Martin Jaggi. Towards fully fp8 gemm llm training at scale. *arXiv preprint arXiv:2505.20524*, 2025.

[10] Nidhal Jegham, Marwan Abdelatti, Chan Young Koh, Lassad Elmoubarki, and Abdeltawab Hendawi. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference, 2025.

[11] Jiedong Lang, Zhehao Guo, and Shuyu Huang. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE, 2024.

[12] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2025.

[13] Asit Mishra, Dusan Stosic, Simon Layton, and Paulius Micikevicius. Recipes for pre-training llms with mxfp8. *arXiv preprint arXiv:2506.08027*, 2025.

[14] Andrei Panferov, Erik Schultheis, Soroush Tabesh, and Dan Alistarh. Quartet ii: Accurate llm pre-training in nvfp4 by improved unbiased gradient estimation. *arXiv preprint arXiv:2601.22813*, 2026.

[15] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

[16] Xingwu Sun, Shuaipeng Li, Ruobing Xie, Weidong Han, Kan Wu, Zhen Yang, Yixing Li, An Wang, Shuai Li, Jinbao Xue, et al. Scaling laws for floating point quantization training. *arXiv preprint arXiv:2501.02423*, 2025.

[17] Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. *arXiv preprint arXiv:2502.20586*, 2025.

[18] Ruizhe Wang, Yeyun Gong, Xiao Liu, Guoshuai Zhao, Ziyue Yang, Baining Guo, Zhengjun Zha, and Peng Cheng. Optimizing large language model training using fp4 quantization. *arXiv preprint arXiv:2501.17116*, 2025.

[19] Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313*, 2024.

[20] 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)*, pages 1–8. IEEE Computer Society, 2025.

[21] Jiecheng Zhou, Ding Tang, Rong Fu, Boni Hu, Haoran Xu, Yi Wang, Zhilin Pei, Zhongling Su, Liang Liu, Xingcheng Zhang, et al. Towards efficient pre-training: Exploring fp4 precision in large language models. *arXiv preprint arXiv:2502.11458*, 2025.

A Supplementary Material

This appendix provides additional implementation and evaluation details for the proposed transposition-invariant FP4 training method. We focus on the details most relevant for reproducibility: training configuration, quantization placement, scaling and rounding rules, evaluation protocol, ablation interpretation, and the scope of the software-emulation results.

B Training Configuration

Table 5 summarizes the training setup used for all model scales. All models are trained with a sequence length of 2K tokens and a token budget of 100B. For each model, the BF16, 2D-FP4, and 2D-FP4+MXFP8 variants use the same optimizer, learning-rate schedule, warmup, and global batch size so that differences in training dynamics are attributable to the numerical format rather than to optimization hyperparameters.

Table 5: Training configurations for all experiments.

Configuration	**OLMo-1B**	**OLMo-7B**	**Qwen 30B MoE**
Model & Data			
Training Tokens	100B	100B	100B
Sequence Length	2K	2K	2K
Objective	Causal LM	Causal LM	Causal LM
Optimization			
Optimizer	AdamW	AdamW	AdamW
Peak Learning Rate	4.0×10^{-4}	3.0×10^{-4}	1.0×10^{-4}
Final Learning Rate	4.0×10^{-5}	3.0×10^{-5}	1.0×10^{-5}
LR Schedule	Cosine Decay	Cosine Decay	Cosine Decay
Warmup Steps	4K	4K	4K
Global Batch Size	2048	2048	512
Weight Decay	0.02	0.02	0.02
Gradient Clipping	1.0	1.0	1.0
Optimizer State Precision	BF16	BF16	BF16
Quantization			
Weight Quantization	2D-FP4	2D-FP4	2D-FP4
Gradient Quantization	2D-FP4	2D-FP4	2D-FP4
Activation Quantization	1D-FP4	1D-FP4	1D-FP4
Query/Key Precision	MXFP8	MXFP8	MXFP8
Default 2D Block Size	32×32	32×32	32×32
Backward Rounding	Stochastic	Stochastic	Stochastic
Scaling Rule	Truncation-free	Truncation-free	Truncation-free

C Quantization Placement

Linear layers. For a linear layer,

$$Y = XW^\top, \qquad \nabla X = \nabla Y W, \qquad \nabla W = (\nabla Y)^\top X.$$

These computations involve transposed tensor views during backpropagation. We therefore apply 2D block quantization to weights and gradient tensors so that the same values retain consistent scale assignments across forward and backward matrix multiplications.

Activations. Activations use a 1D block layout. This avoids grouping semantically unrelated hidden channels into square 2D blocks. In practice, applying 2D blocks to weights and gradients provides the main forward–backward consistency benefit, while 1D activation quantization preserves a simpler channel-aligned layout.

Attention. The query and key projections use MXFP8 because small perturbations in Q and K can be amplified by $QK^\top$ and then by the softmax. The value projection, output projection, and MLP projections use 2D-FP4.

D 2D Block Scaling and Transposition Consistency

Let $X \in \mathbb{R}^{m \times n}$ be partitioned into square blocks of size $b \times b$. For block indices (i, j), define

$$B_{i,j} = X[ib : (i+1)b,\ jb : (j+1)b].$$

The scale for a block is computed from the maximum absolute value in that block:

$$M_{i,j} = \max_{x \in B_{i,j}} |x|.$$

Using truncation-free scaling, the block scale is

$$S_{i,j} = 2^{\left\lceil \log_2\left(\frac{2M_{i,j}}{Q_p - Q_n}\right)\right\rceil},$$

where Q_p and Q_n are the positive and negative representable FP4 bounds. The quantized block is

$$\widehat{B}_{i,j} = S_{i,j} \cdot \mathrm{round}_{\mathrm{FP4}}\left(\frac{B_{i,j}}{S_{i,j}}\right).$$

Under transposition, $B_{i,j}$ maps to a block at the swapped block index:

$$B_{i,j}^\top = X^\top[jb : (j+1)b,\ ib : (i+1)b].$$

The transposed block contains the same values as the original block, so its maximum absolute value is unchanged:

$$M(B_{i,j}) = M(B_{i,j}^\top).$$

Therefore, the scale is also unchanged:

$$S(B_{i,j}) = S(B_{i,j}^\top).$$

This is the transposition-invariance property used by our method. In contrast, a 1D block layout generally changes which values are grouped together after transposition, causing the same value to be quantized with different scales in the forward and backward passes.

Boundary blocks. If a tensor dimension is not divisible by b, the boundary block contains fewer than $b \times b$ valid entries. The scale is computed only over valid entries. Padding values, when used for implementation convenience, are excluded from the scale computation.

E Rounding and Gradient Propagation

During the forward pass, scaled values are rounded to the nearest representable FP4 value. During the backward pass, we use stochastic rounding for gradient tensors. For a value x between adjacent representable values a and b, stochastic rounding chooses a or b with probabilities that preserve the expectation:

$$\mathbb{E}[\text{round}_{\text{stoch}}(x)] = x.$$

This reduces systematic rounding bias in gradient updates. Gradients are propagated through quantization using the straight-through estimator:

$$\frac{\partial \mathcal{L}}{\partial X} \approx \frac{\partial \mathcal{L}}{\partial \widehat{X}}.$$

Thus, quantization is applied in the numerical computation, while the backward derivative of the quantizer is approximated as the identity.

F Evaluation Protocol

Language modeling. We evaluate language modeling quality using perplexity on Wikitext, Pile, and C4. Perplexity is computed as

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{t=1}^{N}\log p(x_t \mid x_{<t})\right),$$

where N is the number of evaluated tokens. Lower perplexity indicates better language modeling performance. Within each model family, BF16 and low-precision variants use the same tokenizer, sequence length, and evaluation batches.

Reasoning tasks. We evaluate downstream reasoning using SciQ, COPA, ARC-Easy, and HellaSwag. Accuracy is computed as

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of evaluated examples}}.$$

For multiple-choice benchmarks, the predicted answer is selected using normalized likelihood over candidate answers. The same scoring procedure is used for BF16, 2D-FP4, and 2D-FP4+MXFP8.

Relative degradation. For perplexity, the relative degradation from BF16 is

$$\Delta_{\text{PPL}}(\%) = 100 \times \frac{\text{PPL}_{\text{method}} - \text{PPL}_{\text{BF16}}}{\text{PPL}_{\text{BF16}}}.$$

For accuracy, we report the absolute difference from BF16:

$$\Delta_{\text{Acc}} = \text{Acc}_{\text{method}} - \text{Acc}_{\text{BF16}}.$$

G Ablation Interpretation

Block size. The default block size is 32×32. Smaller blocks, such as 8×8 and 16×16, reduce quantization error because each scale covers fewer values, but they increase scale metadata and can reduce throughput. Larger blocks, such as 64×64, reduce metadata overhead but increase the final loss gap. The 32×32 setting provides a practical balance between stability, memory overhead, and matrix-multiplication tiling.

Truncation-free scaling. Removing truncation-free scaling makes FP4 training more vulnerable to clipping and overflow. Since FP4 has limited dynamic range, even a small number of clipped high-magnitude values can destabilize training. The ablation in the main paper shows that truncation-free scaling is necessary in addition to the 2D block structure.

Stochastic rounding. Replacing stochastic rounding with deterministic rounding increases bias in gradient tensors. This leads to worse convergence and a larger gap from BF16. The ablation shows that stable FP4 training requires both scale consistency and approximately unbiased gradient quantization.

MXFP8 attention. MXFP8 attention is included to improve numerical robustness in the query/key pathway. The main results show that 2D-FP4 alone is already very close to BF16, while the MXFP8 variant can slightly increase the final perplexity gap in some settings. We therefore view MXFP8 attention as a stability-oriented mixed-precision option, while the primary contribution remains transposition-invariant 2D-FP4 scaling.