HiFloat4 Format for Language Model Pre-training on Ascend NPUs

Mehran Taghian * **Yunke Peng** **Anandharaju Durai Raju** **Xing Huang** **Yao Wang** **Yaoyuan Wang** **Wei Guo** **Yuanyong Luo** **Tianchi Hu** **Junsong Wang** **Xin Wang** **Hu Liu** **Yu Cheng** **Ziwei Yu** **Hongliang Li** **Mehdi Rahimifar** **Lei Yan** **Xuefei Wang** **Zhuang Ma** **Lei Liu** **Hui Yu** **Hoang Le** **Hei Yi Mak** **Tanzila Rahman** **Shadan Golestan**

Huawei

{mehran.taghian1, pengyunke}@huawei.com

Abstract

Training large foundation models at low numerical precision is one of the most promising directions for reducing the compute and memory cost of modern AI. Recent 4-bit floating-point formats such as MXFP4 and NVFP4 can be applied to linear GEMM operations in LLMs, but their limited dynamic range introduces numerical instability that prior work addresses by stacking stabilization mechanisms, typically executed at higher precision and partially eroding the efficiency gains that motivate FP4. In this work, we argue that numerical format design is itself a first-class lever for stable FP4 training, and present the first systematic study of FP4 LLM pretraining on energy-efficient Huawei Ascend NPUs. We compare HiFloat4 (HiF4) against both MXFP4 and NVFP4, holding one recipe fixed across three architectures: dense OpenPangu-1B and Llama3-8B, and Mixture-of-Experts Qwen3-MoE-30B, executing all linear and expert GEMMs in FP4. At matched storage — NVFP4 and HiF4 both spend 4.5 bits per value — the three differ far more in what they need before they will train at all than in final accuracy. HiF4 reaches a relative loss of 1.55% with no stabilization, below fully stabilized MXFP4 (1.79%) and below NVFP4 carrying the per-tensor scaling it cannot train without (2.00%); a single Random Hadamard Transform (RHT) takes it under 1% on the two larger models (0.85%, 0.88%). Our results suggest that stable, accurate FP4 training does not require an ever-growing stack of stabilization techniques; it requires the right numerical format.

1 Introduction

The computational cost of training large language models (LLMs) has become a critical bottleneck. Empirical scaling laws show that performance improves as a power-law function of model size, data, and compute [Kaplan et al., 2020], making it essential to reduce per-step cost so capacity and data can scale without prohibitive hardware and energy demands. Low-precision training is among the most promising directions: representing parameters, activations, and gradients at reduced precision directly cuts memory bandwidth and compute. Prior work has shown that 8-bit floating-point formats can train LLMs effectively when paired with fine-grained block scaling [Micikevicius et al., 2022, Liu et al., 2024], and recent efforts have pushed this frontier to 4-bit formats such as MXFP4 [Rouhani et al., 2023a] and NVFP4 [Abecassis et al., 2025]. Yet FP4's limited representational capacity introduces numerical instability that can derail optimization entirely.

*Author order reflects relative contribution, not seniority or supervisory role.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

A central source of instability is systematic bias in gradient quantization. When gradients contain large outliers, block-level scaling factors are dominated by peak magnitudes, leaving most values confined to a narrow portion of the representable range — and many to underflow to zero — disrupting backpropagation. To compensate, recent work has stacked stabilization techniques: Stochastic Rounding (SR) [Croci et al., 2022], Random Hadamard Transforms (RHT) for gradient decorrelation [Tseng et al., 2025, Abecassis et al., 2025], engineered block-scaling schemes, and other format-specific mechanisms [Cao et al., 2025, Castro et al., 2025, Panferov et al., 2026, Cook et al., 2025]. These auxiliary operations typically run at higher precision (e.g., FP16 or FP32) before FP4 quantization, partially offsetting the efficiency gains that motivated low-precision training. This raises a natural question: rather than patching a format's limitations with increasingly complex stabilization pipelines, can a better-designed numerical representation reduce this burden from the outset?

We argue that numerical format design is itself a first-class lever for stable and efficient FP4 training, and investigate LLM pretraining using the HiFloat4 (HiF4) format [Luo et al., 2026], a 4-bit representation targeting next-generation AI accelerators. HiF4 employs a three-level hierarchical scaling scheme over blocks of 64 elements: an E6M2 top-level scaler, E1 intermediate scalers, and an S1P2 (equivalently E1M2) representation for the FP4 values. By distributing mantissa bits across both global scalers and local elements, this design expands dynamic range while preserving precision — substantially reducing the need for auxiliary stabilization. We evaluate HiF4 against the two FP4 formats with announced hardware support, MXFP4 and NVFP4. NVFP4 is the necessary storage-matched control: its 8-bit E4M3 block scale is amortized over 16 elements, so it costs 4.5 bits per value exactly as HiF4 does, where MXFP4 costs 4.25. At matched storage the accuracy ordering tightens considerably — NVFP4 closes most of the distance between MXFP4 and HiF4, and the margin that remains is 0.01 to 0.14 points, which we do not lean on. What separates the formats is what each needs before it trains at all. HiF4 with no stabilization reaches a lower relative loss than MXFP4 carrying all three of its stabilizers and than NVFP4 carrying the per-tensor scaling (PTS) without which it diverges, under every combination of SR and RHT we tried.

Beyond format comparison, our broader goal is efficient FP4 pretraining on energy-constrained accelerators. We target Huawei Ascend NPUs — domain-specific accelerators with specialized matrix, vector, and scalar units that offer improved power efficiency over general-purpose GPUs — and develop an FP4 training pipeline that executes the majority of linear operations in FP4 while maintaining stable optimization. All experiments run on Ascend hardware, but not on FP4 kernels: no Ascend NPU supports HiF4 or MXFP4 natively yet, so every GEMM executes in BF16 with quantization simulated around it, bit-exactly. We therefore measure the numerical behaviour native kernels would produce, but no throughput, latency or energy. On that basis we evaluate both dense Transformers (OpenPangu-1B, Llama3-8B) and a large Mixture-of-Experts (MoE) architecture (Qwen3-MoE-30B), with $\geq 75\%$ of training computation in simulated FP4 and a loss gap within $\approx 1\%$ of a full-precision baseline.

Our main contributions are:

- We argue that numerical format design is itself a first-class mechanism for stable FP4 training, and that a well-designed representation can reduce the reliance on auxiliary high-precision stabilization pipelines that erode the efficiency gains of low-precision training.
- We investigate HiF4 for pretraining and find that, under matched conditions, a single stabilization technique (RHT on dW) brings its relative loss to 1.25% of the BF16 baseline on OpenPangu-1B and below 1% on Llama3-8B and Qwen3-MoE-30B, with a mean downstream accuracy gap of at most 1.18 points.
- We hold one training recipe fixed across three FP4 formats and find that they differ primarily in stabilization dependence: HiF4 is competitive with no stabilizer at all, MXFP4 needs three, and NVFP4 does not converge without PTS. We do not claim the format itself, which was introduced for inference; what is new is that its hierarchical scaling suffices for stable pretraining, where gradient quantization is the hard part.
- We develop an end-to-end FP4 pretraining pipeline targeting Huawei Ascend NPUs that executes the majority of linear operations in FP4 while maintaining stable optimization on domain-specific, energy-efficient accelerators.
- We validate the pipeline on Dense and MoE transformer models, running $\geq 75\%$ of training computation in FP4 with a loss gap within $\approx 1\%$ of full-precision baselines.

To the best of our knowledge, this is the first systematic study of FP4 LLM pretraining on Ascend NPUs. More broadly, our results suggest that stable, accurate FP4 training does not require an ever-growing stack of stabilization techniques — it requires the right numerical format. HiF4's hierarchical scaling provides enough representational headroom to achieve low relative error on its own; a single addition — RHT to reduce noise in dW — closes the remaining gap, yielding a pipeline that is both simpler and more accurate than prior FP4 approaches. A corollary is that stabilization recipes belong to a format and do not port across formats: SR is necessary for MXFP4, actively harmful for HiF4, and insufficient for NVFP4 at any setting.

2 Related Works

General FP4 Training Several works study 4-bit training without targeting a specific hardware format. LUQ [Chmiel et al., 2021] pairs INT4 weights/activations with FP4 gradients and introduces stochastic underflow to debias subnormal gradients. Zhou et al. [2025] applies per-block FP4 (block size 128) to FFN forward GEMMs only, keeping attention and backward GEMMs in FP8—a mixed-precision rather than end-to-end FP4 scheme. Hu et al. [2025] identifies principles for stable FP4 training: RHT for outlier redistribution, tensor scaling, SR, and a UE5M3 update-scale format (block size 16). Chen et al. [2025a] develops a QSNR theory for block-wise quantization, showing INT can outperform FP depending on crest factor and bit width.

Handling Outliers A complementary line of work attributes low-bit degradation primarily to outliers. Zhang et al. [2025] detects outlier-heavy blocks and falls back to higher precision past a threshold, while Wang et al. [2025] clamps activation outliers and processes the sparse residual in higher precision. Such fallback mechanisms add compute and memory overhead, cause load imbalance, and limit effective FP4 hardware utilization.

Training with MXFP4 Naively applying MXFP4 to all GEMMs is unstable, motivating several stabilization recipes. Tetrajet [Chen et al., 2025b] attributes degradation to biased gradients and forward weight oscillations near quantization thresholds, and addresses them with truncation-free scaling, unbiased double quantization, SR, and the Q-EMA/Q-Ramping stabilizers—at the cost of extra compute and EMA state. Quartet [Castro et al., 2025] decouples forward and backward objectives: Hadamard rotations with RMSE-based clipping (QuEST) preserve effective model capacity in the forward, and SR with RHT preserves effective data efficiency in the backward. Tseng et al. [2025] closes the BF16–MXFP4 gap using SR, RHT, and truncation-free scaling, but quantizes only backward GEMMs; in contrast, we quantize both forward and backward GEMMs and apply RHT selectively to the weight-gradient computation. M2XFP [Hu et al., 2026] augments MXFP with element-level (activations) and subgroup-level (weights) metadata to preserve extreme values, at the cost of additional storage and metadata management.

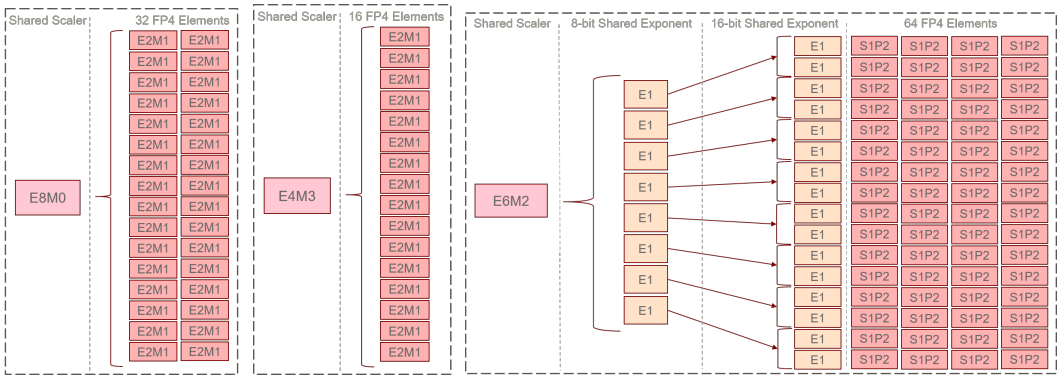
Training with NVFP4 NVFP4 is NVIDIA's Blackwell-supported FP4 format and requires algorithmic support for stable training. Cook et al. [2025] proposes Four-over-Six, mapping block maxima to 4 or 6 to reduce rounding error near the FP4 maximum, at the cost of dual quantization passes. Quartet II [Panferov et al., 2026] introduces MS-EDEN, which combines RHT with SR applied to micro-scale factors rather than individual FP4 values, and uses Nearest Rounding (NR) in the forward and MS-EDEN in the backward to obtain unbiased gradients. NVIDIA's recipe [Abecassis et al., 2025] shows that stable FP4 pretraining at scale requires a combination of selective high-precision layers, SR, RHT, 2D weight quantization for forward–backward consistency, and optional late-stage precision switching. Chen et al. [2025c] improves NVFP4 efficiency by repurposing the redundant negative zero and sign bit of the FP8 scaling factor to encode additional quantization information.

MXFP4 vs. NVFP4 Comparisons Chmiel et al. [2025] shows that fully quantized FP4 LLM training is feasible at scale by combining NVFP4 microscaling (block size 16, E4M3 scale) with selective SR on gradients and backward activations while keeping NR in the forward for stability. Metis [Cao et al., 2025] argues that spectral anisotropy is the dominant obstacle to low-bit training and uses spectral decomposition, adaptive spectral learning rates, and dual-range regularization to match full-precision performance under both MXFP4 and NVFP4.

MoE Models Low-precision MoE training is complicated by routing dynamics and sparse expert activation. Zhang et al. [2026] stores and communicates activations in FP4 but computes in FP8 to maintain stability. Yan et al. [2026] integrates the techniques of Abecassis et al. [2025] into a scalable MoE training framework, but only focuses on system throughput.

Our Work Our work differs from prior FP4 pretraining efforts in three ways. First, where existing studies target general-purpose GPUs, we present the first systematic study of FP4 LLM pretraining on energy-efficient NPU hardware (Huawei Ascend). Second, rather than stacking stabilization mechanisms — SR, RHT, and format-specific scaling schemes, typically executed at higher precision and eroding the very efficiency gains that motivate FP4 — we treat numerical format design as a first-class lever, building on HiF4 [Luo et al., 2026], whose hierarchical scaling provides enough representational headroom that a single stabilization step (RHT on dW) suffices to close the gap to full precision. Third, we validate our approach on both Dense and MoE architectures, executing $\geq 75\%$ of training in FP4 within $\approx 1\%$ of a full-precision baseline. Finally, where the works above evaluate NVFP4 in isolation or against MXFP4, we train it end-to-end under the same pipeline, models, token budget and stabilizer sweep as HiF4 and MXFP4, making it a storage-matched control rather than a cited number.

3 Low-Precision FP4 Training



(a) MXFP4 (b) NVFP4 (c) HiFloat4

Figure 1: FP4 format architectures. (a) MXFP4: 8 bits of shared metadata (E8M0) over 32 4-bit elements, 4.25 bits per value. (b) NVFP4: 8 bits (E4M3) over 16 elements, 4.5 bits per value, plus an FP32 per-tensor scale. (c) HiF4: 32 bits of three-level metadata over 64 elements, also 4.5 bits per value. NVFP4 and HiF4 are therefore storage-matched; MXFP4 is cheaper but coarser.

We employ FP4 computation across all linear layers, including both Dense and MoE components. Specifically, we adopt the recently proposed MXFP4, NVFP4 and HiF4 formats. MXFP4 and HiF4 are candidate formats for next-generation Ascend NPUs and NVFP4 is the storage-matched control (Section 4); no FP4 arithmetic in this paper runs natively on any of them, and all three are emulated (Section 5). All three are used to enable end-to-end FP4 training of large language models (LLMs). MXFP4 [Rouhani et al., 2023b,a] is a 4-bit format using block-wise microscaling: each block of 32 elements is encoded in E2M1 with a shared E8M0 scale. Figure 1a illustrates the architecture. NVFP4 [Abecassis et al., 2025] uses the same E2M1 elements as MXFP4 but a smaller block of 16 with a shared E4M3 scale, layered under a single FP32 per-tensor scale (Figure 1b). Amortizing 8 bits of block scale over 16 rather than 32 elements costs 4.5 bits per value — identical to HiF4, above MXFP4's 4.25 — which makes NVFP4 the storage-matched comparison for HiF4 and MXFP4 the cheaper but coarser one. HiF4 is a hardware-oriented FP4 format for Huawei Ascend NPUs that employs three-level hierarchical scaling. A shared unsigned E6M2 factor provides a coarse block-level scale, while two finer levels of 1-bit micro-exponents refine 8- and 16-way partitions, with each level-3 exponent governing four contiguous FP4 elements. Data elements use S1P2 encoding (equivalent to E1M2). Figure 1c illustrates the architecture. See Appendix B for full details.

Stabilizing training under such extreme quantization demands additional algorithmic techniques beyond standard low-precision arithmetic. To this end, we analyze three state-of-the-art FP4 training frameworks in Table 1, with a focus on their auxiliary high-precision computational overhead beyond FP4 quantization and GEMM operations. For details of how we analyze the computation complexities, see Appendix D. Such auxiliary computations can become a significant bottleneck at scale, diminishing the effective speedup otherwise afforded by low-precision training. Among the three approaches considered—Quartet II [Panferov et al., 2026], Metis [Cao et al., 2025], and

Table 1: Extra compute beyond FP4 GeMM and quantization for methods that apply FP4 in both forward/backward during pretraining. $X \in \mathbb{R}^{M\times K}$, $W \in \mathbb{R}^{N\times K}$, and $D \in \mathbb{R}^{M\times N}$. r is the RHT block size; k, j are Metis's forward/backward randomized SVD ranks. Quartet II's 4/6 forward scale selection adds a $\Theta(MK + NK)$ constant-factor overhead to forward quantization, excluded here. "–" indicates no additional compute beyond baseline GeMM and quantization. The table isolates stabilization cost and excludes the baseline FP4 GEMM and quantization, which every method pays; Quartet II's 4/6 selection is a constant-factor change to double quantization, which belongs to that baseline path, so it falls inside the exclusion as a scoping decision. See Appendix D.

	Forward	Backward	
Method	$Y{=}XW^\top$	$dX{=}DW$	$dW{=}D^\top X$
NVIDIA [Abecassis et al., 2025]	–	–	$(N{+}K)M\log r$
Quartet II [Panferov et al., 2026]	–	$(M{+}K)N\log r$	$(N{+}K)M\log r$
Metis [Cao et al., 2025]	$Mk(K{+}N)$	$MNj{+}jk(M{+}N{+}K)$	$Kj(M{+}N){+}jk(M{+}N{+}K)$

the method proposed by NVIDIA [Abecassis et al., 2025]—the last incurs the lowest computational overhead. We adopt it as our baseline for FP4 training, incorporating the following modifications:

1. 2D weight quantization is omitted. We make no claim about what it would recover and do not estimate its hardware cost; the omission applies identically to all three formats, and we record it in the limitations as an untested deviation from the NVIDIA recipe.
2. Randomized Hadamard Transform (RHT) is applied to the dW computation. Additionally, for MXFP4, we apply SR to the gradient tensor, together with the truncation-free scaling strategy of TetraJet, which adjusts the block scale to keep all values in range rather than clipping out-of-range magnitudes. For NVFP4 we apply SR to the gradient tensor on top of PTS.
3. FP4 training is applied uniformly across all linear layers (except the output and embedding layers), without retaining high-precision computation for selectively sensitive components.

Per-tensor scaling (PTS). NVFP4's E4M3 block scale spans a far narrower exponent range than MXFP4's E8M0 or HiF4's E6M2 plus micro-exponents — 22 binades against HiF4's 69 — so blocks near either edge of it saturate or underflow. PTS corrects this with a single FP32 scale per tensor, derived from the tensor's global maximum and applied before micro-scaling, so that the per-block E4M3 factors encode only variation *within* the tensor. PTS is part of NVFP4's training formulation rather than an optional addition. We nonetheless count it as a stabilization technique throughout this paper, because what we are measuring is the auxiliary high-precision work each format requires outside the FP4 kernel. PTS is a full-tensor reduction and rescale performed in BF16 before quantization, which is the same cost SR, RHT and TF scaling impose. Classifying it by where it runs, rather than by whether the format's authors regard it as optional, is what keeps Table 1 and Section 5.3 consistent with each other and with how we account for the other two formats.

A primary difficulty in extremely low-bit training is how scaled elements are mapped to representable values. NR maps each element to its closest representable value; while hardware-friendly, it introduces a systematic bias that is particularly harmful for gradients, accumulating over iterations. SR introduces controlled randomness prior to rounding and is unbiased in expectation. SR is not uniformly beneficial across formats, however. HiF4's higher per-element mantissa bits and global scaler substantially reduce quantization bias even without SR, so the variance SR introduces outweighs its bias-reduction benefit and degrades performance relative to plain NR. MXFP4 operates at coarser precision, where the bias is more pronounced and SR provides a meaningful correction. As we demonstrate empirically in Section 5.3, SR improves training performance under MXFP4 but is detrimental under HiF4. NVFP4 falls on the MXFP4 side of this divide: it shares the E2M1 element encoding and so the same coarse per-element precision, and SR yields a net benefit once PTS is in place — though SR does not by itself make the format trainable (Section 5.3).

We also apply an RHT to inputs prior to quantization. RHT redistributes energy more uniformly across dimensions, suppressing outliers and reducing the per-coordinate dynamic range; this mitigates underflow of small values in the presence of large outliers and, with SR, reduces the variance of the injected noise. See Appendix C.

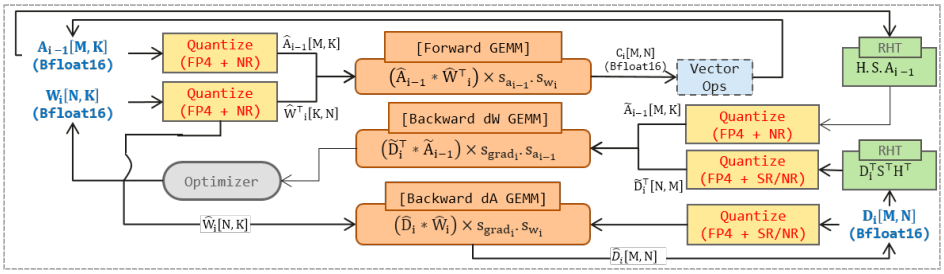


Figure 2: Overview of the GEMM operation within a linear or expert layer. Activations and weights are quantized to HiF4, MXFP4 or NVFP4 using NR; gradients use SR for MXFP4 and NVFP4 and NR for HiF4. An RHT is applied to the weight gradients ($\mathrm{d}W$) prior to quantization. The flow is identical for all three formats; only the quantizer and its stabilizers change.

As illustrated in Figure 2, a linear layer involves three GEMM operations: 1) Forward pass $A_i = \phi(A_{i-1}W_i^\top)$, 2) Backward pass (gradient w.r.t. weight) $dW_i = grad_i^\top A_{i-1}$, 3) Backward pass (gradient w.r.t. input) $dA_{i-1} = grad_i W_i$, where W_i denotes the weight matrix of layer i, A_{i-1} is the input activation from layer $i-1$, grad_i is the upstream gradient propagated from layer $i+1$, and $\phi(\cdot)$ represents the activation function. In the forward GEMM, both activations and weights are quantized to FP4 using NR. During the backward pass, gradient tensors are quantized to FP4 using either NR or SR to mitigate bias accumulation. Empirically, we observe that SR improves training stability and performance for MXFP4 and NVFP4, whereas the opposite is true for HiF4. Furthermore, for the computation of $\mathrm{d}W_i$, RHT with a block size of $k = 64$ is applied along the inner dimension prior to quantization. The transformed tensor is then quantized to FP4, improving numerical stability under low-precision accumulation. The overall workflow of the quantized linear layer is summarized in Figure 2.

4 4-bit Floating Point on Ascend NPUs

Huawei Ascend NPUs [Huawei, 2023] are widely used for large-scale LLM training and inference. Their compute is dominated by the Cube Unit, an array of processing elements (PEs) that perform parallel dot-product and accumulation operations. With a 256-bit input bandwidth, each PE can process a full HiF4 block of 64 FP4 elements in a single dot-product, aligning the HiF4 block size with the PE datapath and minimizing scaling and intermediate reductions. Two properties separate HiF4 from NVFP4 on this datapath. First, one HiF4 block matches the 64-length dot product of a PE, so a single unit pair suffices where NVFP4's 16-element group needs four. Second, applying a scale costs more for NVFP4: HiF4's E6M2 top-level factor needs a floating-point multiply once per 64 elements, its level-2 and level-3 factors reducing to shifts, whereas NVFP4's E4M3 block scale needs one per 16 elements, plus the per-tensor factor PTS introduces. Luo et al. [2026] quantify the consequence at six fewer multipliers, roughly a third of the incremental area, and about 10% less power. We need to emphasize that we measure no area, power or throughput anywhere in this paper. We make no equivalent claim against MXFP4, whose power-of-two E8M0 scale requires no floating-point multiplier at all and which is likewise well matched to the Ascend datapath.

5 Experimental Analysis

Our experiments aim to quantify the FP4 and BF16 pretraining gap. Concretely, we investigate:

1. *Pretraining dynamics.* We compare loss curves under BF16, HiF4, MXFP4, and NVFP4, and measure their relative deviation throughout training.
2. *Downstream quality.* We evaluate pretrained models on downstream tasks to assess how low-precision training affects downstream performance.
3. *Sensitivity analysis.* We ablate the design choices that maintain stability and accuracy in the low-precision regime.

Table 2: Percentage of compute executed in FP4, broken down by component

	Attention		FFN		Experts		
Model	QKV	Output	FC1	FC2	Up	Down	Total
OpenPangu-1B	10.61%	5.30%	42.45%	21.22%	–	–	79.58%
Llama3-8B	8.86%	5.91%	41.37%	20.68%	–	–	76.82 %
Qwen3-MoE-30B	3.13%	2.50%	–	–	60.18%	30.09%	95.90%

Emulation. No FP4 GEMM in this paper runs natively. All GEMMs execute in BF16 on Ascend 910B3 NPUs with quantization and dequantization emulated around them at bit-exact precision, so the loss curves are what native hardware would produce for the same recipe; what is unmeasurable here is speedup, not accuracy, and we report no throughput, latency or energy figures. All runs share hardware, software, seeds and data order and differ only in the quantization path. Training is performed on a cluster of 64 nodes, each equipped with 8 NPUs (65 GB of memory per NPU), 192 CPU cores, and 1536 GB of host memory.

Models. We study three architectures: two dense Transformers, OpenPangu-1B and Llama3-8B, and one mixture-of-experts (MoE) model, Qwen3-MoE-30B. In the dense models, linear layers reside in the attention block (QKV and output projections) and the FFN (FC1 and FC2). These linear components account for $\approx 80\%$ of the parameters in OpenPangu-1B and $\approx 77\%$ in Llama3-8B. For Qwen3-MoE-30B, we apply FP4 to the QKV and output projections together with the up- and down-projections inside each expert. The attention projections contribute only $\approx 5\%$ of total compute, whereas the experts hold $\approx 90\%$ of model parameters.

FP4 coverage. Table 2 reports the fraction of compute executed in FP4 for each model component. FP4 is applied uniformly to every linear layer except output and embedding, for all three formats. This is stricter than NVIDIA's recipe, which keeps roughly 15% of linear layers in higher precision and reports divergence when every one is quantized [Abecassis et al., 2025], so our NVFP4 results are measured under a harsher regime than that work uses. Low-precision training is particularly advantageous for MoE: although the experts hold $\approx 90\%$ of parameters, top-8 routing over 128 experts activates only 6.25% of experts per forward/backward pass. As a result, only $\approx 16\%$ of the model's parameters participate in computation at any given step, of which $\approx 75\%$ reside in linear layers and are executed in FP4. The remaining $\approx 84\%$ of parameters, corresponding to inactive experts, can potentially be *stored* in FP4 without incurring any additional compute. FP4 thus delivers a double benefit on MoE models: memory savings on the dormant majority of weights, and throughput gains on the active subset.

Training setup. We use a unified optimizer configuration across all models: 50B tokens with Adam under a cosine learning-rate schedule, global batch size 2048 for OpenPangu and 512 for Llama and Qwen, with 480 and 2,000 warmup steps respectively. Full hyperparameters are in Appendix A.

5.1 Pretraining Dynamics

Figure 3 compares the three FP4 schemes against the BF16 baseline, each in its own best configuration. HiF4 tracks BF16 at least as closely as the best-tuned MXFP4 and NVFP4 recipes while requiring the fewest stabilization techniques: HiF4 uses only RHT, MXFP4 additionally requires TF-scaling and SR, and NVFP4 additionally requires SR on top of the PTS. We attribute this to the higher per-element mantissa precision of HiF4 and its hierarchical scaling, which reduces the need for auxiliary stabilization the other two formats require to recover representational capacity.

To quantify the gap, we compute the relative loss error between each low-precision run and the BF16 baseline: $\mathrm{E}_{\mathrm{Relative}} = (\mathrm{loss}_{\text{low-precision}} - \mathrm{loss}_{\mathrm{baseline}})/\mathrm{loss}_{\mathrm{baseline}}$. Table 3 reports this quantity for the curves in Figure 3. HiF4 achieves lower relative error than both NVFP4 and MXFP4, and on Llama3-8B and Qwen3-MoE-30B its gap to BF16 falls below 1%. The comparison against NVFP4 is much tighter than against MXFP4. Ordered by active parameter count rather than total parameters — OpenPangu-1B (1B), Qwen3-MoE-30B (4.6B), Llama3-8B (8B) — the HiF4 relative error decreases monotonically from 1.25% to 0.88% to 0.85%, while its advantage over NVFP4 grows from 0.01 to 0.11 to 0.14 points. The gap between the two formats therefore widens as model scales. We note

that this comparison treats dense and MoE models on the same axis; total MoE parameters may still contribute to representational capacity in ways that active count alone does not capture.

The Llama3-8B HiF4 run (Figure 3, top middle) shows two visible loss spikes, and the 0.85% we report is from that run rather than a clean rerun. Both occur before roughly 10B tokens; thereafter the loss rejoins the BF16 trajectory for the remaining 40B. Spikes of this kind appear in all our runs, the BF16 reference included; HiF4's are the most visible but recover rather than diverge. The MXFP4 configuration on the same model shows fewer spikes, indicating that the additional TF-scaling and SR stabilizers dampen transient instabilities, at the cost of higher steady-state loss and per-step overhead. Section 5.3 presents an ablation study on FP4 stabilization methods, showing that the configuration used here is the most effective.

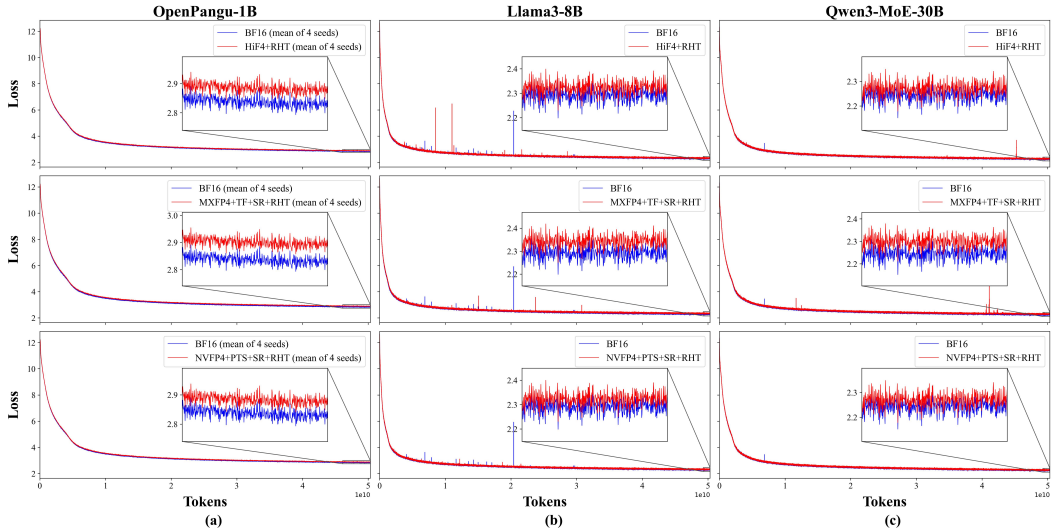


Figure 3: Training loss curves for OpenPangu-1B (a), Llama3-8B (b) and Qwen3-MoE-30B (c) over 50B tokens, each FP4 scheme against BF16. Top: HiF4+RHT. Middle: MXFP4+TF+SR+RHT. Bottom: NVFP4+PTS+SR+RHT. OpenPangu-1B is averaged across 4 seeds; insets show the late-training region where the gap is most visible. HiF4 tracks BF16 at least as closely as MXFP4 and NVFP4 on all three models despite using fewer stabilizers.

Table 3: Relative loss error with respect to the BF16 baseline across three models trained on 50B tokens, ordered by active parameter count. Each format uses its own best configuration (RHT for HiF4; TF+SR+RHT for MXFP4; PTS+SR+RHT for NVFP4, see Section 5.3). OpenPangu-1B is averaged across 4 seeds; the larger models are single runs without error bars.

Model	Activate Parameters	MXFP4	NVFP4	HiFloat4
OpenPangu-1B	1B	1.75%	1.26%	**1.25%**
Llama3-8B	8B	1.44%	0.99%	**0.85%**
Qwen3-MoE-30B	4.6B	1.55%	0.99%	**0.88%**

5.2 Downstream Performance

We evaluate the three pretrained models on tasks spanning science knowledge, reading comprehension, commonsense reasoning, and language modeling. Specifically, we report 5-shot accuracy on AI2 Reasoning Challenge–Easy (ARC-E) and Challenge (ARC-C) [Clark et al., 2018], HellaSwag [Zellers et al., 2019], PIQA [Bisk et al., 2020], SciQ [Welbl et al., 2017], BoolQ [Clark et al., 2019], and WinoGrande [Sakaguchi et al., 2021], plus WikiText perplexity [Merity et al., 2016], comparing each FP4 format against the BF16 baseline. Results are summarized in Table 4.

The downstream results track the loss results and add little to them. HiF4 has the smallest mean accuracy gap on both dense models (-0.71 and -0.24, against -1.59/-0.71 for MXFP4 and

$-1.41/-1.29$ for NVFP4); on Qwen3-MoE-30B the three fall within 0.3 points of one another, with NVFP4 nominally ahead, which we do not read as separating them. On WikiText, HiF4 and NVFP4 are effectively tied and both clearly ahead of MXFP4: the increase over BF16 is at most $+0.71$ for HiF4 and $+0.69$ for NVFP4 against $+1.03$ for MXFP4, with HiF4 and NVFP4 never differing by more than 0.03 on any model. Both of the 4.5-bit formats see their perplexity penalty roughly halve from 1B to 30B while MXFP4's never falls below $+0.60$, which mirrors Table 3.

Table 4: Comparison of HiF4, NVFP4, and MXFP4 against the BF16 baseline across three pretrained models. BF16 rows report absolute 5-shot accuracy (%), averaged over three seeds, and WikiText perplexity (PPL; lower is better, single run). FP4 rows report the *change* relative to the BF16 row of the same model, with red indicating degradation and blue improvement. Mean is computed over the seven accuracy benchmarks only. The best FP4 result per model is shown in bold.

Model	Quant	ARC-E	ARC-C	HellaS	PIQA	SciQ	BoolQ	WinoG	Mean	PPL
OpenPangu-1B	BF16	47.47	26.42	35.12	65.23	80.97	60.94	51.70	52.55	28.87
	MXFP4	-1.53	-0.66	-1.11	-0.56	-2.13	-3.39	-1.79	-1.59	+1.03
	NVFP4	-1.65	-0.40	-0.75	+0.11	-0.43	-6.19	-0.58	-1.41	**+0.69**
	HiF4	+0.28	-0.52	-0.88	-1.18	-2.47	-0.75	+0.53	**-0.71**	+0.71
Llama3-8B	BF16	67.32	36.89	59.03	74.43	93.47	57.80	58.01	63.85	15.19
	MXFP4	-1.08	-2.78	-1.79	-0.47	+0.07	+4.04	-2.97	-0.71	+0.60
	NVFP4	-1.81	-2.41	-0.78	+0.73	-0.40	-1.90	-2.47	-1.29	**+0.36**
	HiF4	-1.72	-1.62	-0.83	-0.74	+0.17	+4.96	-1.87	**-0.24**	+0.38
Qwen3-MoE-30B	BF16	68.81	36.66	61.16	75.75	93.97	59.80	58.33	64.93	14.45
	MXFP4	-0.92	+1.57	-1.99	-0.56	-0.24	-3.55	-2.79	-1.21	+0.78
	NVFP4	+0.62	+2.28	-0.76	+0.02	+0.36	-7.23	-1.63	**-0.91**	**+0.31**
	HiF4	-1.65	+0.14	-1.08	-0.47	-0.24	-2.36	-2.61	-1.18	+0.34

BoolQ and ARC-C are by a wide margin the noisiest benchmarks in the suite, and both move by several points in either direction across formats and models. On Llama3-8B, BoolQ gains $+4.96$ for HiF4 and $+4.04$ for MXFP4, while on OpenPangu-1B and Qwen3-MoE-30B it loses -6.19 and -7.23 for NVFP4. ARC-C falls for every format on both dense models, then rises on Qwen3-MoE-30B for MXFP4 ($+1.57$) and NVFP4 ($+2.28$) but not for HiF4 ($+0.14$). Since perplexity rises for every FP4 format on every model, we read the positive swings on both tasks as evaluation noise rather than genuine modeling improvement: a benchmark that moves down at two scales and up at the third, with no consistent ordering among the formats, is reporting seed variance. The negative swings under NVFP4 on BoolQ are one-sided, however, and that one task accounts for much of its mean accuracy gap; we flag it as a caveat rather than a conclusion, since one binary-choice benchmark cannot separate a format-level effect from seed variance.

5.3 Sensitivity Analysis

We ablate the sensitivity of the three formats to the stabilization techniques available to each. TF scaling does not apply to HiF4 or NVFP4, whose scaling designs avoid the truncation issue it addresses, while PTS does not apply to MXFP4 or HiF4, whose scalers already cover the required dynamic range. Figure 4a reports relative loss vs. BF16 for all applicable combinations on OpenPangu-1B, while Figure 4b shows key configurations on Qwen3-MoE-30B.

The formats differ substantially in how much stabilization they require. HiF4 without stabilization already reaches 1.55%, outperforming MXFP4 with its full stabilization stack (1.79%) and approaching NVFP4 with PTS and RHT (1.42%). Adding RHT further reduces HiF4 to 1.19%, the best result across all configurations. Thus, HiF4 reaches its best performance with a single technique, whereas MXFP4 and NVFP4 benefit from stacking multiple techniques.

NVFP4 shows a particularly strong dependence on PTS: without it, all configurations diverge, including those with SR, RHT, or both. This is primarily due to its limited dynamic range rather than rounding: RHT redistributes energy and SR reduces rounding bias, but neither changes the representable range. NVFP4 covers only 22 binades, compared with 69 for HiF4. Luo et al. [2026] report a similar failure at inference time, where direct-cast NVFP4 collapses and PTS restores performance. With PTS, RHT provides the largest improvement ($2.00 \rightarrow 1.42\%$), while SR gives a further 0.15 percentage-point gain.

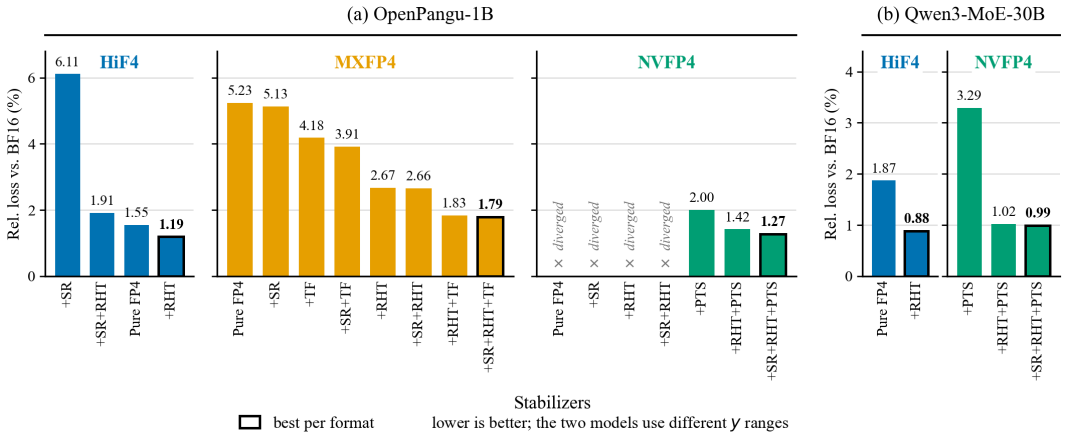


Figure 4: Ablation of FP4 stabilizers on (a) OpenPangu-1B and (b) Qwen3-MoE-30B, both trained for 50B tokens. Bars show loss relative to the BF16 baseline; lower is better. NVFP4 diverges without PTS. The Qwen3-MoE-30B runs cover HiF4 and NVFP4 only, testing whether the 1B ordering holds at scale. Best configuration per format is outlined and in bold. Ablation runs are single seed.

These trends persist on Qwen3-MoE-30B. HiF4 without stabilization reaches 1.87%, compared with 3.29% for NVFP4 with PTS alone. RHT reduces NVFP4 loss by 2.27 percentage points and HiF4 loss by 0.99 points. For HiF4, RHT is the only beneficial technique: SR substantially hurts performance, both alone and when combined with RHT. We hypothesize that HiF4's scaling already preserves the small-magnitude information that SR is intended to recover. For MXFP4, RHT provides the largest single-technique gain (2.56 points), while TF provides a further 1.05 points; RHT+TF (1.83%) nearly matches the full configuration (1.79%). Overall, HiF4 requires less stabilization and is uniquely sensitive to the choice of technique.

6 Conclusion and Future Directions

We present a systematic study of end-to-end FP4 pretraining for large language models on Ascend NPUs, comparing HiFloat4 (HiF4) against MXFP4 and NVFP4 under one fixed recipe. We claim that the hierarchical scaling is enough on its own for stable pretraining, where gradient quantization is the difficulty inference work never meets. HiF4 and NVFP4 end up close on accuracy – HiF4 still better. What separates them is how much high-precision help each needs first: HiF4 is competitive with none, while MXFP4 and NVFP4 each need three mechanisms, and NVFP4 does not converge at all without PTS. A stabilization recipe therefore belongs to a format and does not port between formats — SR is necessary for MXFP4, harmful for HiF4, and on its own not enough for NVFP4.

Limitations. Our comparison is fixed at 50B tokens, and nothing here speaks to convergence over long horizons. Abecassis et al. [2025] report instabilities appearing late in much longer runs and caution that small-scale conclusions may not transfer; since their gaps also widen with tokens, our absolute numbers likely understate a long run. All FP4 arithmetic is emulated, so we evaluate numerical behaviour only; for HiF4 no silicon exists yet. Only OpenPangu-1B is multi-seed, and the 0.01–0.14 point HiF4–NVFP4 margins are not separable from run-to-run variation on this evidence, which is why the paper rests on the stabilization results instead. We omit 2D weight quantization, an untested deviation from the NVIDIA recipe applied equally to all three formats, and we do not study spending MXFP4's 0.25-bit saving on higher-precision sensitive layers — a budget HiF4 could spend the same way. Our ablation also establishes the stabilization ordering only as an early-horizon property, which is where divergence happens.

Future Directions. Extending FP4 to post-training, especially RL fine-tuning, is unexplored and high-impact, as are long-context and multimodal training. The long-horizon study this paper cannot provide becomes affordable, and far more useful, once native HiF4 silicon exists: convergence and speedup could then be measured in the same run.

References

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.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

Hengjie Cao, Mengyi Chen, Yifeng Yang, Ruijun Huang, Fang Dong, Jixian Zhou, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, et al. Metis: Training llms with fp4 quantization. *arXiv preprint arXiv:2509.00404*, 2025.

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.

Mengzhao Chen, Meng Wu, Hui Jin, Zhihang Yuan, Jing Liu, Chaoyi Zhang, Yunshui Li, Jie Huang, Jin Ma, Zeyue Xue, et al. Int vs fp: A comprehensive study of fine-grained low-bit quantization formats. *arXiv preprint arXiv:2510.25602*, 2025a.

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

Yuzong Chen, Xilai Dai, Chi-chih Chang, Yash Akhauri, and Mohamed S Abdelfattah. The power of negative zero: Datatype customization for quantized large language models. *arXiv preprint arXiv:2501.04052*, 2025c.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. *arXiv preprint arXiv:2112.10769*, 2021.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, and Song Han. Four over six: More accurate nvfp4 quantization with adaptive block scaling. *arXiv preprint arXiv:2512.02010*, 2025.

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3), 2022.

Robert Hu, Carlo Luschi, and Paul Balanca. Elucidating the design space of fp4 training. *arXiv preprint arXiv:2509.17791*, 2025.

Weiming Hu, Zihan Zhang, Haoyan Zhang, Chen Zhang, Cong Guo, Yu Feng, Tianchi Hu, Guanglin Li, Guipeng Hu, Junsong Wang, et al. M2xfp: A metadata-augmented microscaling data format for efficient low-bit quantization. *arXiv e-prints*, pages arXiv–2601, 2026.

Huawei. *Huawei Atlas AI Computing Solution*, pages 163–219. Springer Nature Singapore, Singapore, 2023. ISBN 978-981-19-2879-6. doi: 10.1007/978-981-19-2879-6_6. URL `https://doi.org/10.1007/978-981-19-2879-6_6`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Yuanyong Luo, Jing Huang, Yu Cheng, Ziwei Yu, Kaihua Zhang, Kehong Hong, Xinda Ma, Xin Wang, Anping Tong, Guipeng Hu, et al. Hifloat4 format for language model inference. *arXiv preprint arXiv:2602.11287*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

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.

Bita Darvish Rouhani, Nitin Garegrat, Tom Savell, Ankit More, Kyung-Nam Han, Mathew Zhao, Ritchie Hall, Jasmine Klar, Eric Chung, Yuan Yu, Michael Schulte, Ralph Wittig, Ian Bratt, Nigel Stephens, Jelena Milanovic, John Brothers, Pradeep Dubey, Marius Cornea, Alexander Heinecke, Andres Rodriguez, Martin Langhammer, Summer Deng, Maxim Naumov, Paulius Micikevicius, Michael Siu, and Colin Verrilli. Ocp microscaling (mx) specification. Technical report, Open Compute Project, 2023a.

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*, 2023b.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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

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.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, 2017.

Zijie Yan, Hongxiao Bai, Xin Yao, Dennis Liu, Tong Liu, Hongbin Liu, Pingtian Li, Evan Wu, Shiqing Fan, Li Tao, Robin Zhang, Yuzhong Wang, Shifang Xu, Jack Chang, Xuwen Chen, Kunlun Li, Yan Bai, Gao Deng, Nan Zheng, Vijay Anand Korthikanti, Abhinav Khattar, Ethan He, Soham Govande, Sangkug Lym, Zhongbo Zhu, Qi Zhang, Haochen Yuan, Xiaowei Ren, Deyu Fu, Tailai Ma, Shunkang Zhang, Jiang Shao, Ray Wang, Vasudevan Rengasamy, Rachit Garg, Santosh Bhavani, Xipeng Li, Chandler Zhou, David Wu, Yingcan Wei, Ashwath Aithal, Michael Andersch, Mohammad Shoeybi, Jiajie Yao, and June Yang. Scalable training of mixture-of-experts models with megatron core, 2026. URL `https://arxiv.org/abs/2603.07685`.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

Pengle Zhang, Jia Wei, Jintao Zhang, Jun Zhu, and Jianfei Chen. Accurate int8 training through dynamic block-level fallback. *arXiv preprint arXiv:2503.08040*, 2025.

Wuyue Zhang, Chongdong Huang, Chunbo You, Cheng Gu, Fengjuan Wang, and Mou Sun. Practical fp4 training for large-scale moe models on hopper gpus. *arXiv preprint arXiv:2603.02731*, 2026.

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 Training Configuration

Table 5: Training configurations for all experiments.

Configuration	**OpenPangu-1B**	**Llama3-8B**	**Qwen3-MoE-30B**
Model & Data			
Training Tokens	50B	50B	50B
Sequence Length	4K	4K	4K
Optimization			
Optimizer	AdamW	AdamW	AdamW
Start Learning Rate	10^{-4}	10^{-4}	10^{-4}
End Learning Rate	10^{-5}	10^{-5}	10^{-5}
LR Schedule	Cosine Decay	Cosine Decay	Cosine Decay
Warmup Steps	480	2K	2K
Global Batch Size	2048	512	512

B Low-Precision Formats

B.1 MXFP4

MXFP4 [Rouhani et al., 2023b,a] is a 4-bit floating-point format proposed within the Open Compute Project (OCP) framework. It adopts a block-wise microscaling design in which each tensor is partitioned into blocks of 32 elements. Within each block, individual values are encoded using a private E2M1 representation (2 exponent bits and 1 mantissa bit), while a shared E8M0 scaling factor provides a common block-level scale. The E2M1 format supports representable values in $\pm\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$, whereas the shared E8M0 scaler spans a dynamic range of $\left[2^{-127}, 2^{127}\right]$. The effective representable range of each block is therefore determined by the product of the shared E8M0 scale and the local E2M1 values, enabling substantial dynamic range expansion while maintaining compact 4-bit storage. Figure 1a illustrates MXFP4 architecture. To mitigate information loss from out-of-range values exceeding ± 6, we adopt a truncation-free scaling strategy similar to TetraJet [Chen et al., 2025b]. Instead of clipping large magnitudes to the representable E2M1 limits, the block-level scaling factor is adjusted to ensure that all values remain within range, thereby avoiding severe distortion introduced by hard truncation.

B.2 NVFP4

NVFP4 [Abecassis et al., 2025] is NVIDIA's Blackwell-supported 4-bit floating-point format. Like MXFP4 it encodes individual values in E2M1, with representable magnitudes $\pm\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$, but it differs in two respects. First, the block size is 16 rather than 32, so the shared scale adapts to a quarter as many values. Second, the shared scale is an E4M3 floating-point factor rather than a power-of-two exponent, which gives the block scale a mantissa and therefore finer granularity, at the cost of a much narrower exponent range: E4M3 spans roughly $[2^{-9}, 2^{8}]$ against E8M0's $[2^{-127}, 2^{127}]$. This narrower range is why NVFP4 requires per-tensor scaling (Appendix C.3): a single FP32 factor per tensor is applied before micro-scaling so that the E4M3 block scales only need to cover variation within the tensor. Storage-wise, 8 bits of block metadata amortized over 16 elements gives an overhead of 0.5 bits per value and a total of 4.5 bits per value — identical to HiF4 and above MXFP4's 4.25 — which makes NVFP4 the storage-matched comparison point for HiF4 throughout this paper. Figure 1b illustrates the NVFP4 architecture.

B.3 HiFloat4

HiFloat4 (HiF4) is a hardware-oriented FP4 format designed for Huawei Ascend NPUs, featuring a three-level hierarchical scaling scheme to improve both precision and dynamic range while maintaining 4-bit storage per value. At the first level, HiF4 employs a shared unsigned 8-bit E6M2 floating-point scale, allocating 6 bits to the exponent (bias 48) and 2 bits to the mantissa with an implicit leading 1. The value is represented as $X_{E6M2} = 2^E \times 1.M$. This level provides a coarse global scaling factor for each block. The second and third levels introduce fine-grained scaling through 1-bit micro-exponents, denoted as $E1_8$ and $E1_{16}$, corresponding to 8-way and 16-way hierarchical partitioning, respectively. In this structure, the level-1 E6M2 factor serves as the global base scale shared across the block. Each level-2 micro-exponent modulates a subset of the block and is further refined by two adjacent level-3 micro-exponents. Each level-3 micro-exponent governs four contiguous 4-bit data elements within its local group, enabling localized dynamic range adaptation. The metadata for all three scaling levels occupies 32 bits per block of 64 elements, resulting in an amortized overhead of 0.5 bits per value. The 4-bit data elements are encoded in S1P2 format, where S denotes the sign bit, "1" represents the integer bit preceding the binary point, and "2" corresponds to two fractional bits. This representation is equivalent to an E1M2 floating-point format. Figure 1c illustrates the HiF4 architecture. For a detailed architectural description and design rationale, we refer the reader to [Luo et al., 2026].

C Stabilization Techniques for Low-Precision Training

C.1 Unbiased Quantized Training with Stochastic Rounding

Training with low-bit formats requires mapping scaled values to a small set of representable points. The standard choice, NR, deterministically selects the nearest representable value and thereby introduces a systematic quantization bias. This bias is particularly harmful for gradients, where it accumulates across iterations and can degrade optimization dynamics. To mitigate this, we adopt SR, which introduces controlled randomness prior to rounding. Concretely, for a value x, we sample $\Delta \sim \mathcal{U}(-\delta, \delta)$ and compute $\hat{x} = \mathrm{Round}(x + \Delta)$, where $\mathrm{Round}(\cdot)$ denotes rounding to the nearest representable value in the target 4-bit format, and $\delta = 0.5$ controls the noise magnitude. By construction, SR yields unbiased quantization in expectation, eliminating the systematic error inherent to NR.

C.2 Low-Variance Quantized Training with Random Hadamard Transform

To further stabilize low-precision quantization, we apply an RHT to the input tensors prior to quantization. The transform redistributes energy more uniformly across dimensions, suppressing outliers and reducing the per-coordinate dynamic range. This mitigates underflow of small values in the presence of large outliers and, when combined with SR, reduces the variance of the noise that SR injects. Formally, given an input tensor $X \in \mathbb{R}^{j \times k}$, the transform is defined as $\tilde{X} \leftarrow HSX$, where $S = \mathrm{diag}(d_1, \ldots, d_k) \in \mathbb{R}^{k \times k}$ is a random sign matrix with $d_i \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}\{\pm 1\}$, and $H \in \mathbb{R}^{k \times k}$ is the normalized Hadamard matrix, defined recursively as

$$H_n = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}, \qquad H_1 = [1]. \tag{1}$$

Since H and S are both orthogonal, we have $HS(HS)^\top = I$. Consequently, inserting RHT into a GEMM preserves the exact matrix product $A^\top B = (HSA)^\top (HSB)$, ensuring mathematical equivalence in full precision while improving quantization robustness under low-precision arithmetic.

C.3 Per-Tensor Scaling

Block-wise formats assume that the shared scale can represent the magnitude of the block it governs. For MXFP4 and HiF4 this assumption is safe: E8M0 spans $[2^{-127}, 2^{127}]$ and HiF4's E6M2 top-level scale, combined with two levels of 1-bit micro-exponents, covers the dynamic range of any tensor encountered in practice. NVFP4's E4M3 block scale does not: its exponent range is narrow enough that, for tensors whose global magnitude sits away from unity, whole blocks saturate at the top of the scale range or underflow at the bottom. PTS removes this failure mode with a second, coarser level

of scaling. A single FP32 factor s_T is computed per tensor from its global maximum and divided out before micro-scaling, so the per-block E4M3 factors encode only relative variation within the tensor; s_T is reapplied to the GEMM output. The extra cost per quantization is one global reduction over the tensor and one scalar multiply per element, both negligible against the GEMM itself, but the effect on stability is not marginal: as Section 5.3 shows, every NVFP4 configuration without PTS diverges, regardless of how much SR or RHT is applied. PTS is thus a precondition for NVFP4 training rather than a refinement of it, which is why Section 5.3 treats it as part of the NVFP4 baseline; it has no counterpart in MXFP4 or HiF4.

D Analyzing Computation Complexity of Metis and Quartet II

D.1 NVIDIA's recipe

NVIDIA stabilizes NVFP4 backward passes with a single tool: an RHT applied along the contracted dimension of each backward GeMM. We assume that the rotation uses a Walsh-Hadamard structure, applying it to a tensor of T elements costs $T\log(r)$, where r is the rotation block size. Each backward GeMM has two operands that need rotating, so the total extra cost per backward pass is roughly (matrix perimeter) × log r. The forward pass adds nothing — it relies on 2D weight quantization to keep weights reusable across forward and backward, avoiding any extra rotation work.

D.2 Quartet II

Quartet II uses the same basic tool — Hadamard rotations on the backward inner dimension — but adds an unbiasedness correction (MS-EDEN) on top of round-to-nearest quantization, and re-quantizes weights on the backward pass instead of reusing them. At the asymptotic level, the rotation cost is identical to NVIDIA's: (matrix perimeter) $\times \log(r)$ per backward GeMM. The MS-EDEN correction adds a small inner-product computation per rotation block, but this is sub-leading and gets absorbed into the rotation cost. On the forward pass, Quartet II's 4/6 scale selection roughly doubles per-element quantization work, but quantization is $\Theta(matrixsize)$ and dwarfed by the GeMM, so this doesn't show up in the asymptotic table either.

D.3 Metis

Metis takes a fundamentally different approach and pays a fundamentally different price. Instead of rotating to smooth distributions, it decomposes matrices into a low-rank "dominant" piece and a full-rank "residual" piece, then quantizes each separately. This adds three distinct sources of overhead:

Forward path through low-rank factors. The forward pass now has two routes: a baseline-cost route through the residual matrix, and an extra low-rank route through the rank-k factors. The low-rank route costs rank × (matrix perimeter) — concretely $Mk(K+N)$ for input X and rank-k factors of W.

Per-step gradient decomposition. This is the term with no analog elsewhere. Every backward pass, Metis freshly decomposes the gradient matrix D via randomized SVD, which costs roughly MNj where j is the gradient rank. Notice that this term scales with the product MN of two large dimensions — unlike RHT, which only scales with the sum of dimensions times $\log(r)$. This is the dominant cost in Metis and the conceptual reason its overhead is heavier than the rotation-based methods.

Mixed contractions. When the decomposed gradient meets the decomposed weights in computing parameter updates, the resulting chains of small matrix products cost roughly jk × (sum of all three matrix dimensions). The rank parameters k and j are tunable, which is both a feature and a complication. The paper sets them to 1% of matrix rank for FP8 (where Metis is essentially free) but 50% of matrix rank for FP4 (where Metis is the headline result). At 50%, the MNj term alone becomes comparable to the baseline GeMM cost — Metis's stabilization adds roughly 50% on top of the original compute, versus a few percent for the rotation-based methods. This is the price of treating anisotropy structurally rather than smoothing it away.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the contributions and scope through accurate empirical analysis.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section 6

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the information necessary to reproduce the main experimental results. We describe the low-precision training framework, model architecture, mixed-precision configuration, training setup, and optimization hyperparameters used in all experiments. These details cover the aspects that directly affect the main claims regarding convergence and numerical behavior.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We are unable to release code as our implementation is part of an internal confidential LLM pretraining framework built for Ascend NPUs. To support reproducibility, we provide detailed algorithmic descriptions, hyperparameter settings, and training configurations in the paper and supplementary material, enabling reimplementation of our method in standard GPU-based training frameworks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all training and evaluation details necessary to understand our results. Section 3 presents our quantization framework, with stabilization techniques detailed in Appendix C and quantization formats in Appendix B. Section 5 describes model architectures, evaluation benchmarks, and evaluation protocols, while training hyperparameters are reported in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars, confidence intervals, or significance tests. We do average over multiple seeds wherever the compute cost permits. The OpenPangu-1B pretraining results in Table 3 and Figure 3 are the mean of 4 seeds, which differ in

initialization and data order, and the downstream accuracies in Table 4 are the mean of 3 evaluation seeds per 5-shot run. Repeating pretraining at the scale of Llama3-8B and Qwen3-MoE-30B is prohibitively expensive, so those are single runs, and the stabilizer sweeps in Figure 4 are single seed. We therefore report means without a measure of spread. We state this limitation in Section 6, where we also note that the 0.08–0.14 point HiF4–NVFP4 margins cannot be separated from run-to-run variation on this evidence; our conclusions rest instead on the stabilization-dependence results, where the differences between formats are far larger than any plausible seed effect.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 reports the hardware used for our experiments (Ascend NPUs), including the number of devices, details about the cluster, cpu, and memory. Additionally, memory requirements can be followed directly from the reported model sizes, quantization formats, and training configuration, which together determine the parameter and activation memory footprint.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work focuses on low-precision quantization techniques for LLM pretraining, a foundational efficiency contribution that is not tied to any particular downstream application or deployment. Quantization reduces the compute, memory, and energy required to train large models, which has the positive impact of lowering the resource barrier for LLM research and reducing environmental costs. Like any general-purpose efficiency improvement for LLMs, it could indirectly enable both beneficial and harmful applications of language models, but it does not introduce new capabilities or risks beyond those already inherent to LLM pretraining itself.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper focuses on quantization method for LLM pretraining and does not release pretrained model weights, datasets, or any other artifacts that pose a high risk of misuse. As such, no safeguards for responsible release are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original sources of all datasets, benchmarks, and method used in our experiments. The evaluation benchmarks were publicly released under permissive licenses and used for their intended purpose of evaluating language models. Any open-source libraries used in our implementation are credited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets such as datasets, pretrained models, or code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are the subject of study in this work (we develop quantization techniques for LLM pretraining), but they are not used as a component of the research methodology itself. Any use of LLMs by the authors was limited to writing and editing assistance, which does not require declaration under the NeurIPS LLM policy.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.