FP4 All the Way: Fully Quantized Training of LLMs

Brian Chmiel [^*†] **Maxim Fishman** [^*†] **Ron Banner** [^†] **Daniel Soudry** [°]

[^]Nvidia, Israel

[°]Department of Electrical and Computer Engineering - Technion, Haifa, Israel

{brianchmiel, maxim.fishman.d, ronbanner.RB, daniel.soudry}@gmail.com

Abstract

We demonstrate, for the first time, fully quantized training (FQT) of large language models (LLMs) using predominantly 4-bit floating-point (FP4) precision for weights, activations, and gradients on datasets up to 1T tokens. We extensively investigate key design choices for FP4, including block sizes, scaling formats, and rounding methods. Our analysis shows that the NVFP4 format, where each block of 16 FP4 values (E2M1) shares a scale represented in E4M3, provides optimal results. We use stochastic rounding for backward and update passes and round-to-nearest for the forward pass to enhance stability. Additionally, we identify a theoretical and empirical threshold for effective quantized training: when the gradient norm falls below approximately $\sqrt{3}$ times the quantization noise, quantized training becomes less effective. Leveraging these insights, we successfully train a 7-billion-parameter model on 256 Intel Gaudi2 accelerators. The resulting FP4-trained model achieves downstream task performance comparable to a standard BF16 baseline, confirming that FP4 training is a practical and highly efficient approach for large-scale LLM training. A reference implementation is supplied in `https://github.com/Anonymous1252022/fp4-all-the-way`.

1 Introduction

The rapid advancement of Large Language Models (LLMs) has led to unprecedented breakthroughs in natural language understanding and generation. State-of-the-art models now scale to hundreds of billions of parameters, enabling remarkable capabilities across diverse applications. However, this progress comes at a significant cost, with training and inference demanding immense computational power and memory bandwidth. As model sizes grow, hardware constraints become a major bottleneck, necessitating innovations in numerical precision and memory-efficient architectures.

Until recently, the dominant numerical format for pretraining Large Language Models (LLMs) was BF16, which provided a balance between precision and efficiency. However, as model sizes and dataset scales have grown, researchers have explored lower-precision alternatives to improve computational efficiency and reduce memory requirements. A few pioneering studies have demonstrated that full training in FP8 is not only feasible but also effective. [13] showcased the potential of FP8 training on small-scale datasets (100B),[8] extended it to trillions of tokens dataset with a 7B parameters model, and [6] demonstrated FP8's viability at an even larger scale, successfully training a massive 671B parameter Mixture-of-Experts (MoE) model on a vast dataset, achieving state-of-the-art results. FP8 is quickly emerging as the new standard for large-scale LLM training. As research continues to push precision boundaries further, the next logical step is exploring FP4, which promises even greater efficiency while maintaining training stability and model accuracy.

[*]Equal contribution

[†]Work done while at Intel.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Recently, NVIDIA introduced its new GPU architecture, Blackwell [1], marking a major milestone as the first GPU to support FP4 matrix multiplication in hardware. This advancement enables nearly 2× acceleration in matrix multiplication throughput compared to FP8, significantly boosting performance and energy efficiency for large-scale deep learning workloads. Blackwell supports two distinct FP4 formats—MXFP4 and NVFP4—each with different design choices in block size and scale encoding. In this paper, we aim to systematically investigate the full range of block sizes and scaling formats for FP4 training, analyzing their impact on accuracy and stability and highlighting the advantages of the NVFP4 format.

Beyond the choice of numerical format and block size, another critical aspect of low-precision quantization is the rounding mode used during quantization. The two most commonly used rounding modes are round-to-nearest (RtN), which deterministically maps each value to its closest representable value, and stochastic rounding (SR), which probabilistically rounds based on the distance to neighboring values to reduce quantization bias. While prior empirical work [19, 4] has highlighted the importance of SR, particularly in the backward pass for stabilizing gradient updates, its benefits have largely been observed heuristically. In this work, we provide an analysis of noisy gradients training and identify the point where they stop being effective and show the importance of SR in low-precision training.

Recent studies have explored the use of FP4 formats for training large language models, yet none achieved full quantization across all key components—weights, activations, and gradients. In contrast, our approach introduces, for the first time, a fully quantized FP4 training framework that addresses all these components simultaneously, and on significantly larger datasets. Table 2 highlights these important distinction. This end-to-end FP4 training setup enables us to evaluate the practical viability of low-precision methods at scale. Moreover, we evaluate our FP4 models on various downstream tasks—showing on-par results with the BF16 baseline—demonstrating that efficiency gains do not come at the cost of quality.

We make several key contributions:

1. **FP4 Format Optimization.** We conduct comprehensive experiments on block size and scaling formats for FP4 training, revealing that block sizes below 16 elements offer diminishing returns on accuracy. Our comparison of different exponent-mantissa configurations (E1M6 through E8M0) found the best performance was achieved by E4M3—the format used in NVFP4, validating NVIDIA's hardware design choices.
2. **Split Rounding Strategy.** We establish that combining stochastic rounding in the backward pass with round-to-nearest in the forward pass significantly improves training stability and final model accuracy in FP4 training compared to using either method alone throughout the training process.
3. **Precision Transition Analysis.** We provide a theoretical framework that identifies the critical point at which FP4 precision becomes less effective for continued training progress. Specifically, when the full-precision gradient standard deviation falls below $\sqrt{3}$ times the quantization noise standard deviation. We suggest, at the end of the training, to apply a higher precision Quantization Aware Finetuning (QAF) to increase the signal-to-noise ratio higher above this threshold and quickly converge to the baseline.
4. **End-to-End Large-Scale FP4 Training.** We successfully train a 7B-parameter LLM entirely in FP4 precision on a trillion tokens, using 256 Intel Gaudi2 accelerators. While a slight gap in final training loss is observed compared to the BF16 baseline, it is fully closed through the brief QAF phase. This leads to downstream task performance on par with BF16, confirming the practical viability of FP4 for real-world large-scale applications.

2 Related work

Quantization is a key area of research in the effort to compress neural networks for more efficient deployment and reduced resource consumption. The two predominant approaches in this space are Post-Training Quantization (PTQ) [22, 11, 9] and Quantization-Aware Training (QAT) [7, 3, 12]. PTQ focuses on converting pre-trained models to low-bit representations without additional training, making it attractive for rapid deployment, especially in inference scenarios. In contrast, QAT incorporates quantization effects during model training or fine-tuning, enabling the network to adapt and maintain accuracy under low-precision constraints. Both methods have seen significant

advancements, with recent work demonstrating competitive performance at 4-bit precision [9] and below [20].

Fully Quantized Training (FQT) is a more challenging task than PTQ or QAT, as it requires training from scratch with low-precision weights, activations, and gradients to accelerate all matrix multiplications. Until recently, applying FQT beyond 16-bit precision was considered difficult due to instability and convergence issues. However, recent works have demonstrated its feasibility. [13] presented the first FQT of a large language model in FP8 on a dataset of up to 100 billion tokens. [8] extended this to 2 trillion tokens, revealing stability issues in later training stages and proposing a modified activation function to address them. [6] further advanced the field by training a large Mixture-of-Experts (MoE) model with FP8 FQT, mitigating instability through finer-grained quantization.

Finer granularity quantization is emerging as a key direction for enabling Fully Quantized Training (FQT) beyond FP8, particularly in the context of FP4 precision. By reducing the quantization block size, these methods aim to better capture local variations in data distributions, improving stability and accuracy. Notable examples include MXFP4 [15] and NVFP4 [1].

The two works most closely related to ours are [21, 19]. [21] proposes training large language models using a vector-wise FP4 format combined with two key techniques: a Differentiable Gradient Estimator (DGE) to replace the standard Straight-Through Estimator (STE), and Outlier Clamp Compensation (OCC) to handle activation outliers wih an additional sparse residual matrix. Their models are trained on up to 100 billion tokens, but they quantize only weights and activations, keeping gradients in higher precision—thus accelerating only one of the three matrix multiplications involved in training. [19], in contrast, focuses on gradient quantization using the MXFP4 format and applies stochastic rounding alongside the Hadamard transform to stabilize training. They train models up to 40 billion tokens. However, like [21], they only accelerate part of the three matrix multiplications. In contrast, our work is the first to demonstrate full FP4 Fully Quantized Training (FQT) of large-scale LLMs, enabling the acceleration of all matrix multiplications during training.

3 FP4 training

Going beyond FP8 to FP4 training presents significant challenges due to the limited dynamic range of FP4, making it difficult to capture the full variability of activations and gradients without excessive quantization error. However, the recently introduced microscaling floating-point family (MXFP) [15] offers a promising alternative by dynamically adjusting the scale at finer granularity, mitigating precision loss.

MXFP4, includes 1 sign bit, 2 exponent bits, and 1 mantissa bit (E2M1) is a floating-point format that enhances low-precision training by dividing data into blocks of size 32, with each block sharing a common scale. The scale for each block is stored using the E8M0 format, an 8-bit exponent-only representation that provides a wide dynamic range without a mantissa and sign. Another potential format for FP4 training is NVFP4, which uses the same E2M1 data representation as MXFP4 but differs in block size and scaling format. NVFP4 divides data into smaller blocks of size 16, compared to MXFP4's 32, allowing for finer-grained scaling adjustments. Additionally, NVFP4 employs an E4M3 format for storing scales, providing a balance between dynamic range and precision. Both MXFP4 and NVFP4 are supported in NVIDIA's Blackwell architecture [1]. In Table 1 we compare these 2 formats.

Table 1: Comparison of MXFP4 and NVFP4 Formats

Datatype	**MXFP4**	**NVFP4**
Data Representation	E2M1	E2M1
Block Size	32	16
Scale Format	E8M0	E4M3
Per-Tensor Scale	No	Yes

3.1 Exploring block size and scale format

Seeing the differences between the two FP4 formats supported in Blackwell—MXFP4 and NVFP4—in terms of block size and scale format, we decided to investigate their impact further. Specifically, we aim to compare the full range of possible scale formats, while maintaining the FP8 data format. Additionally, we explore different block sizes to understand their effect on numerical stability, training efficiency, and model accuracy. This analysis will provide deeper insights into the trade-offs between dynamic range, precision, and computational efficiency in low-precision training.

In Fig. 1 we train a 350M Llama-style model with FP4 format (E2M1) with block size 16 and different scaling formats. Note that, similar to NVFP4, most configurations (except for E8M0, which corresponds to MXFP4) do not utilize the sign bit in the scale. This may represent a potential inefficiency that future work could aim to exploit. We notice that the best results are achieved with E3M4 and E4M3, where the latter is used in the NVFP4 format. In Fig. 2 we compare different block sizes both with scales formats E8M0 and E4M3, which are the scales used in MXFP4 and NVFP4, respectively. Note that while selecting the appropriate scale has a significant impact on final accuracy—for example, E1M6 leads to complete divergence—the block size has a more modest effect, with smaller block sizes generally yielding better results. This leads us to proceed with the NVFP4 format, which uses a block size of 16 and a scale format of E4M3.

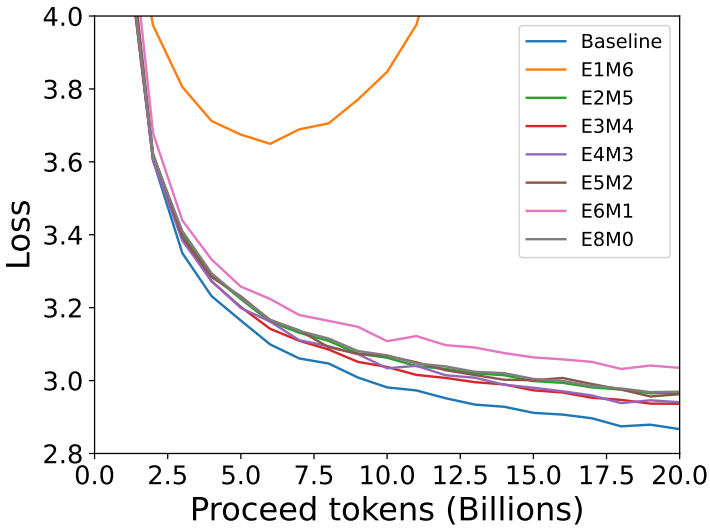


Figure 1: **Formats E4M3 (used in NVFP4) and E3M4 achieved the best results.** Comparison of different scaling formats (E1M6, E2M5, E3M4, E4M3, E5M2, E6M1, E8M0) when training a 350M Llama model using FP4 format (E2M1) with block size 16. The formats E3M4 and E4M3 achieve the best results (recall E4M3 is used in NVFP4), whereas E1M6 results in complete divergence.

3.2 Exploring the rounding modes

After completing our exploration of the various block sizes and scales within the FP4 format, we now turn our attention to another critical aspect: the choice of rounding modes. In the next section, we will delve into the different rounding strategies available for FP4 and analyze their impact on numerical stability and model performance.

Fully quantized training encompasses the quantization of three key general-matrix-multiplications (GEMMs): forward, backward, and update. Each GEMM involves two quantized operands—resulting in six distinct quantization points across the training pipeline:

$$\textbf{[Forward]} \quad z_l = Q(W_l)Q(a_{l-1}); \qquad a_l = f_l(z_l) \tag{1}$$

$$\textbf{[Backward]} \quad g_{l-1} = Q(W_l^T)Q(\delta_l); \quad \delta_l = f_l'(z_l) \odot g_l \tag{2}$$

$$\textbf{[Update]} \quad \frac{\partial C}{\partial W_l} = Q(\delta_l)Q(a_{l-1}^T)\,, \tag{3}$$

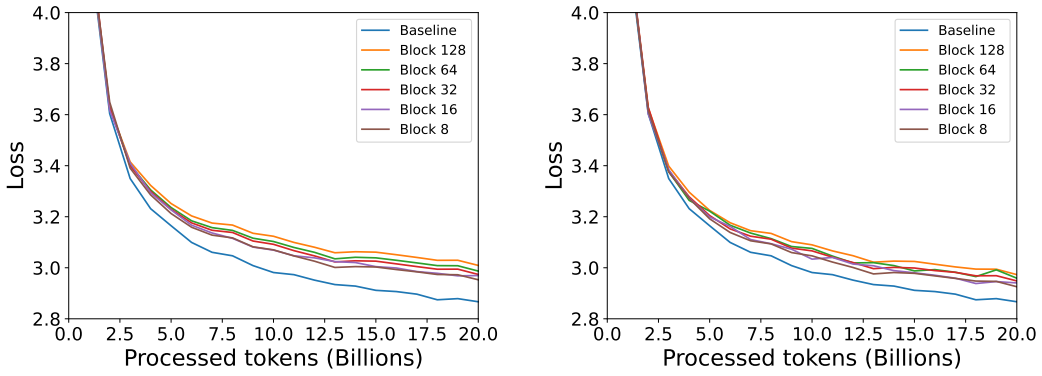


Figure 2: **Block size 16 is the best option.** We examine the impact of different block sizes (8, 16, 32, 64, 128) on training accuracy using scaling formats: (a) E8M0 (used in MXFP4) and (b) E4M3 (used in NVFP4). Smaller block sizes yield modest improvements in accuracy, with diminishing returns below 16 elements per block. Thus, a block size of 16 provides an optimal compromise between performance and computational overhead.

where C is the loss function, Q is a quantization operation, $\odot$ is a component-wise product and, in each layer l, f_l is the activation function, W_l is weight matrix, z_l are the pre-activations, and $g_l \triangleq \frac{\partial C}{\partial a_l}$.

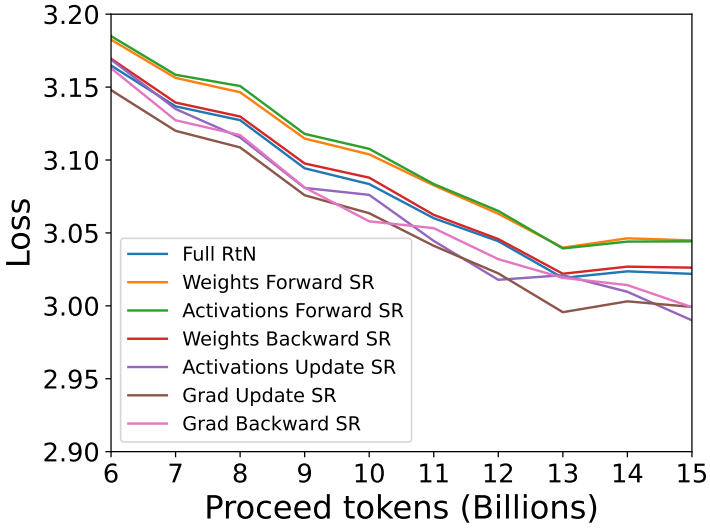


Figure 3: **Comparison of different rounding schemes when training a 350M Llama model using NVFP4 format.** In each graph, we apply SR in one of the six elements in one of the GEMMs while the rest use round-to-nearest (RtN). Notice that applying SR to neural gradients during both 'Update' and 'Backward' GEMMs and activations during the 'Update' GEMM leads to lower training loss, while applying SR to other components has the opposite effect, increasing the loss.

Notably, we have the flexibility to select the rounding mode independently for each of these six elements. In Fig. 3, we present results of training a 350M Llama model using NVFP4 where stochastic rounding (SR) is applied to each of these elements separately while the rest of the elements use round-to-nearest (RtN), allowing us to evaluate its individual contribution. Applying SR to neural gradients during update and backward GEMMs and activations during the update GEMM helps reduce training loss, whereas using SR in other components leads to an increase in loss. In Fig. 7 in the Appendix, we present additional experiments that support the same conclusion. As a result, we

adopt the following selective rounding scheme:

$$\textbf{[Forward]} \quad z_l = Q_{\mathrm{RtN}}(W_l)Q_{\mathrm{RtN}}(a_{l-1}); \qquad a_l = f_l(z_l) \tag{4}$$

$$\textbf{[Backward]} \quad g_{l-1} = Q_{\mathrm{RtN}}(W_l^T)Q_{\mathrm{SR}}(\delta_l); \quad \delta_l = f_l'(z_l) \odot g_l \tag{5}$$

$$\textbf{[Update]} \quad \frac{\partial C}{\partial W_l} = Q_{\mathrm{SR}}(\delta_l)Q_{\mathrm{SR}}(a_{l-1}^T)\,, \tag{6}$$

4 Analysis of Quantized SGD with Stochastic Rounding

This section analyzes when training with low-precision gradients stops being effective when using stochastic rounding (SR). While SR removes bias and enables stable descent during much of training, its benefits diminish once gradients become too small relative to quantization noise. We derive a threshold on the gradient-to-noise ratio that signals when further progress stalls, motivating a precision switch for backward and update passes. This switch improves convergence without altering the forward pass or the deployed model.

Key takeaways from the analysis:

- **SR enables unbiased updates**, allowing stable descent even under aggressive FP4 quantization. In contrast, deterministic rounding introduces a persistent bias, leading to an irreducible error floor and preventing convergence (see Appendix B.2).
- **There exists a critical threshold:** With SR, we show in Section 4.1 (and in more detail in Appendix B.1) that the average per-coordinate gradient magnitude falls approximately below $\sqrt{3}$ times the quantization noise standard deviation, training no longer yields effective loss reduction. This threshold is derived under simplifying assumptions—e.g., gradient descent with optimal step size, Taylor approximation of the loss, and a concentrated Hessian spectrum.
- **Empirical evidence supports the theory:** in Section 4.2, for both synthetic and real-model settings, we show empirically performance degrades sharply below this threshold. A precision switch guided by the theory restores convergence.

4.1 Theoretical Derivation

We begin with a second-order Taylor expansion of the loss function around the current parameter vector θ_t, which reveals a descent term proportional to $-\nabla L^T \Delta\theta$ and a curvature term involving the Hessian H. We then replace the full-precision gradient ∇L with its quantized version $g_q = \nabla L + \varepsilon$, where ε is zero-mean noise introduced by stochastic rounding.

Using the update rule $\Delta\theta = -\eta g_q$ and taking expectations under SR ($\mathbb{E}[\varepsilon] = 0$, $\mathbb{E}[\varepsilon\varepsilon^T] = \sigma_q^2 I$), we obtain the expected loss change:

$$\mathbb{E}[\Delta L] = -\eta \left\|\nabla L\right\|^2 + \tfrac{1}{2}\eta^2 \left(\nabla L^T H\, \nabla L + \sigma_q^2 \operatorname{tr}(H)\right).$$

Balancing the descent and noise terms yields the optimal step size:

$$\eta^* = \frac{\|\nabla L\|^2}{\nabla L^T H\, \nabla L + \sigma_q^2 \operatorname{tr}(H)}.$$

Substituting η^* back into the expected loss gives:

$$\mathbb{E}[\Delta L] = -\frac{\|\nabla L\|^4}{2\left(\nabla L^T H\, \nabla L + \sigma_q^2 \operatorname{tr}(H)\right)}.$$

Differentiating with respect to σ_q, and using some simplying assumptions reveals that sensitivity to quantization noise peaks when:

$$\sigma_{\mathrm{critical}} = \frac{\|\nabla L\|}{\sqrt{3d}}.$$

Once the per-coordinate gradient magnitude drops below $\sqrt{3}\,\sigma_q$, the descent becomes negligible and higher-precision gradients are needed to continue improving the loss.

4.2 Empirical Validation

To validate this theoretical threshold, we present two types of experiments.

First, we simulate training on a simple quadratic loss with an adaptive noise schedule. We scale the quantization noise to $\sigma_q = k \cdot \sigma_{\text{critical}}$ for $k = 2, 1, 0.5$. As shown in Fig. 4, convergence completely stalls at high noise levels (e.g., $k = 2$), slows near the critical threshold ($k = 1$), and closely tracks full-precision training when noise is reduced below the threshold ($k = 0.5$).

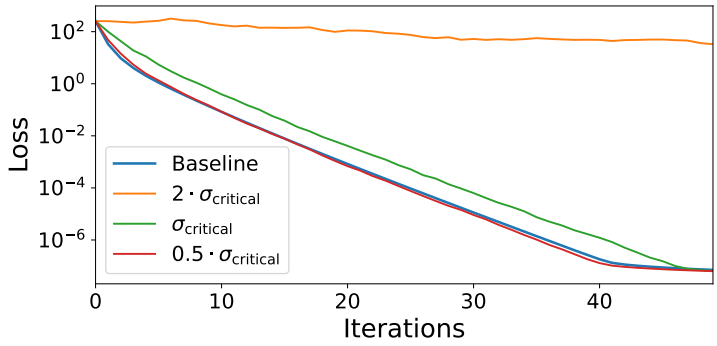


Figure 4: **Validation of theoretical derivation in a simple quadratic loss.** Training loss with noise levels $\sigma_q = k \cdot \sigma_{\text{crit}}$ for $k = 2, 1, 0.5$ in a toy quadratic model. High noise blocks descent; low noise allows continued progress.

Second, we test in Fig. 5, the threshold on a real 60M-parameter Llama model. During training, we monitor the ratio $\|\nabla L\|/(\sigma_q\sqrt{d})$, and switch to higher-precision gradients at the 1000th iteration. When this ratio crosses $\sqrt{3}$, the loss gap to the full-precision baseline closes immediately, validating the predictive power of the threshold.

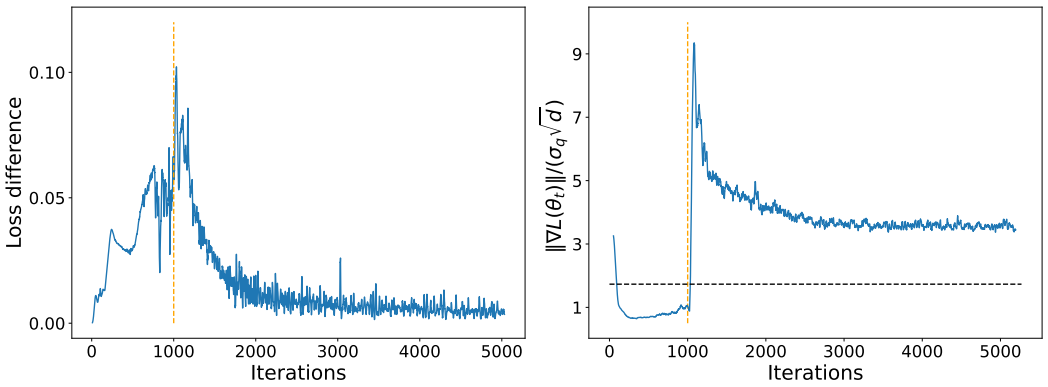


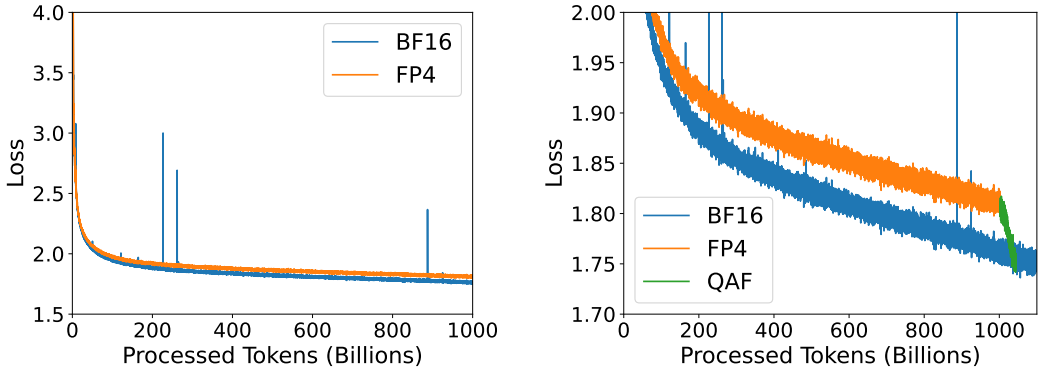
Figure 5: **Validation theoretical prediction in a Llama 60M model. (Left):** The difference between the loss curve of the baseline and a model with increasing precision mid-training (1000th iteration, vertical dashed orange line). After increasing the precision, the loss difference is completely reduced. **(Right):** Gradient-to-noise ratio with the $\sqrt{3}$ threshold (black dashed line).

5 Experiments

Setup. We used the Llama2 model [18] as our baseline. This model is a decoder-only Transformer [2] with pre-normalization RMSNorm [23], Smooth-SwiGLU activation function [8], and rotary positional embeddings [17]. We trained the models on the open-source Red Pajama dataset [5] for $1T$ tokens, maintaining hyperparameters consistent with [18], including train-test split and initialization. Specifically, we used AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$. We used cosine learning rate schedule, with 2000 steps of warmup, peak learning rate of 3×10^{-4} and decay to 0.1 of the peak

learning rate. We used a global batch-size of 4M tokens. All training was conducted on 256 Intel Gaudi2 devices, during $\sim$ 30 days.

FP4 training. In Fig. 6a we present our main experiment and present the training loss of Llama2 7B with the proposed FP4 scheme, which includes the use of the NVFP4 format (block size 16, scale format E4M3) and applying SR in the neural gradients (update + backward GEMMs) and activations (update GEMM), while applying RtN for the weights (forward + backward GEMMs) and activations (forward GEMM). In Table 2 we compare the quantization settings of our work with two previous FP4 training works [19, 21], showing we are the first work that allows the acceleration of all matrix multiplication during training. In the Appendix Table 4 we show the similarity of the FP4 training losses at different seeds, showing the noise robustness of the proposed method.



(a) (b)

Figure 6: **Full FP4 training a 7B model achieved same loss as BF16 baseline. (a):** Training loss of Llama2 7B using the proposed FP4 scheme which include NVFP4 format (block size 16, scale format E4M3) with SR in the neural gradients (update + backward GEMMs) and activations (update GEMM). Notice a small gap in training loss is observed. (Table 3). **(b):** Quanization aware finetuning (QAF) training loss of Llama2 7B using NVFP4 format in the forward GEMM and BF16 in the backward and update GEMMs. Notice this short QAF is able to completely close the gap with BF16 baseline.

Table 2: Comparison of the different FP4 training works. DGE refers to Differentiable Gradient Estimator, OCC to Outlier Clamp Compensation, SR to Stochastic Rounding and RHT to Random Hadamard Transform. While previous works quantize only part of the GEMMs to FP4 with additional overhead of residual sparse matrix (OCC [21]) or Hadamard transform (RHT [19]), our work is the first work that show quantization to FP4 of all GEMMs operands, without adding significant overhead.

	Weight	**Activation**	**Neural gradients**	**Tokens**
[21]	FP4+DGE+OCC	FP4+DGE+OCC	BF16	100B
[19]	BF16	BF16	MXFP4+RHT+SR	21B
Ours	NVFP4 (RtN)	NVFP4 (RtN / SR)	NVFP4 (SR)	1T

Quantization Aware Fine-tuning (QAF). FP4 training results (Fig. 6a) in a small gap in training loss compared to the BF16 baseline. As shown in Fig. 5, increasing the precision raises the gradient-to-noise ratio higher above the critical threshold. To close the remaining gap, we introduce a brief quantization-aware finetuning (QAF) phase, where pretraining continues on the same dataset with the forward pass kept in FP4, while the backward pass is executed in BF16. Keeping the forward path in FP4 ensures that the model remains fully compatible with low-precision inference, without requiring any additional processes such as post-training-quantization. During this stage, we reset the learning rate and apply a short warmup (40 iterations) followed by a cosine decay schedule with an initial peak learning rate. In Fig. 6b we present that this short QAF can completely close the gap with BF16 baseline, achieving an average bits of 4.3 bits for GEMMs across training + QAF. In the Appendix

Table 5 we show an ablation study of the QAF length ratio required to get similar final loss as BF16. We find it decreases for larger datasets.

Zero-shot Performance. Table 3 compares the zero-shot performance (accuracy and perplexity) on downstream tasks between the BF16 baseline and our FP4 model — both after the FP4 training phase (1T tokens) and after QAF phase (+40B tokens). Notice that while a small gap is observed in part of the tasks after the FP4 training phase, it is completely closed after the QAF.

Table 3: Zero shot accuracy and perplexity comparison between the BF16 baseline and the proposed FP4, both after the FP4 training phase (1T) and after QAF phase (+40B). HS refers to HellaSwag, WG refers to Winogrande, BQ refers to BoolQ, AC refers to Arc-C, PQ refers to PiQA, LA refers to Lambada, GQ refers to GPQA. IE refers to If Eval. MB refers to MBPP, TQ refers to TrivialQA. XS refers to XSum. Notice that after the QAF, the proposed FP4 model achieved on-par results with the BF16 baseline.

Precision	Data	Accuracy ↑												Perplexity ↓	
		LA	HS	WG	AC	BQ	PQ	GQ	IE	MB	TQ	XS	Avg	Wiki	LA
BF16	1T	61.52	68.71	66.54	38.14	69.33	76.33	24.54	33.81	8.2	34.99	11.97	45.63	5.54	6.1
BF16	1.04T	61.46	68.57	64.48	38.91	70.09	75.41	24.73	30.94	8.6	34.19	12.33	45.13	5.54	6
FP4	1T	58.39	67.31	64.01	38.65	69.33	74.86	24.73	32.13	6.4	34.3	12.09	44.46	5.83	6.77
+ QAF	+40B	67.71	68.53	65.98	39.25	68.9	76.01	27.29	32.25	9.4	38.31	12.11	45.75	5.56	5.97

6 Discussion

This work presents the first demonstration of fully quantized FP4 training—covering weights, activations, and gradients—at large scale. Our experiments on Llama2 7B model show that, while a small gap in training loss initially appears compared to BF16, this gap can be fully closed with a short QAF phase, where the forward pass remains in FP4 and only the backward pass switches to BF16. Importantly, downstream task performance remains on par with BF16, confirming FP4's practical viability.

A key contribution is our investigation of FP4 format design. We find that NVFP4 (E4M3 with block size 16) offers the best trade-off between dynamic range and precision. Other blocks size or alternative exponent/mantissa configurations lead to instability or diminishing returns, aligning with NVIDIA Blackwell's hardware decisions.

We also introduce a split rounding strategy, using stochastic rounding only in the backward pass, which substantially improves training stability. Furthermore, our theoretical analysis identifies a critical transition point: when the full-precision gradient standard deviation falls approximately below $\sqrt{3}$ times the quantization noise, training stagnates. This insight guides the design of the final fine-tuning phase to boost the signal-to-noise ratio and match BF16 convergence.

Limitations. A key limitation of this work is the lack of dedicated FP4 support in current Gaudi hardware, which prevents us from directly measuring the potential speedup and energy efficiency benefits of native FP4 execution. As a result, all experiments are conducted using FP4 simulations in Gaudi2, which incur additional overhead from precision casting and lead to longer runtimes. Based on previous FP8 works [13, 8] we expect in a rough estimation to $\sim 35-40\%$ time-to-train acceleration in comparison to FP8, which corresponds to $\sim 85\%$ time-to-train acceleration in comparison to the BF16 baseline. Our work centers on the LLaMA architecture, one of the most widely adopted frameworks in modern LLMs. Preliminary experiments indicate that the approach can be directly applied to Mixture-of-Experts (MoE) architectures. Further analysis of MoE models and extensions to vision tasks are reserved for future work.

Acknowledgements

The research of DS was Funded by the European Union (ERC, A-B-C-Deep, 101039436). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency (ERCEA). Neither the European Union nor the granting authority can be held responsible for them. DS also acknowledges the support of the Schmidt Career Advancement Chair in AI.

References

[1] Nvidia blackwell architecture. URL https://resources.nvidia.com/en-us-blackwell-architecture.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[3] Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi. Liu. Optimize weight rounding via signed gradient descent for the quantization of llms. *ArXiv*, abs/2309.05516, 2023. URL https://api.semanticscholar.org/CorpusID:261697282.

[4] Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. In *International Conference on Learning Representations*, 2021. URL https://api.semanticscholar.org/CorpusID:246634451.

[5] Together Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

[6] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jun-Mei Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shao-Ping Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wen-Xuan Yu, Wentao Zhang, X. Q. Li, Xiangyu Jin, Xianzu Wang, Xiaoling Bi, Xiaodong Liu, Xiaohan Wang, Xi-Cheng Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yao Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yi-Bing Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxiang Ma, Yuting Yan, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Zehui Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437, 2024. URL https://api.semanticscholar.org/CorpusID:275118643.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314, 2023. URL https://api.semanticscholar.org/CorpusID:258841328.

[8] Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. Scaling fp8 training to trillion-token llms. *ArXiv*, abs/2409.12517, 2024. URL https://api.semanticscholar.org/CorpusID:272753088.

[9] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv*, abs/2210.17323, 2022. URL https://api.semanticscholar.org/CorpusID:253237200.

[10] Amirata Ghorbani, James Wexler, and Been Kim. Automating interpretability: Discovering and testing visual concepts learned by neural networks. *ArXiv*, abs/1902.03129, 2019. URL https://api.semanticscholar.org/CorpusID:59842921.

[11] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *GetMobile Mob. Comput. Commun.*, 28:12–17, 2023. URL https://api.semanticscholar.org/CorpusID:258999941.

[12] Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L. Castro, Mahdi Nikdan, and Dan Alistarh. Quest: Stable training of llms with 1-bit weights and activations. *ArXiv*, abs/2502.05003, 2025. URL https://api.semanticscholar.org/CorpusID:276236108.

[13] Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. Fp8-lm: Training fp8 large language models. *ArXiv*, abs/2310.18313, 2023. URL https://api.semanticscholar.org/CorpusID:264555252.

[14] Jeffrey Pennington and Pratik Worah. Nonlinear random matrix theory for deep learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/0f3d014eead934bbdbacb62a01dc4831-Paper.pdf.

[15] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verilli, Ralph Wittig, Doug Burger, and Eric S. Chung. Microscaling data formats for deep learning. *ArXiv*, abs/2310.10537, 2023. URL https://api.semanticscholar.org/CorpusID:264146384.

[16] Levent Sagun, Utku Evci, V. Ugur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ArXiv*, abs/1706.04454, 2017. URL https://api.semanticscholar.org/CorpusID:35432793.

[17] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), mar 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

[18] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez,

Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL https://api.semanticscholar.org/CorpusID:259950998.

[19] Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. *ArXiv*, abs/2502.20586, 2025. URL https://api.semanticscholar.org/CorpusID:276725175.

[20] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *ArXiv*, abs/2310.11453, 2023. URL https://api.semanticscholar.org/CorpusID:264172438.

[21] 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*, abs/2501.17116, 2025. URL https://api.semanticscholar.org/CorpusID:275932373.

[22] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *ArXiv*, abs/2211.10438, 2022. URL https://api.semanticscholar.org/CorpusID:253708271.

[23] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

.

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 claim of the paper is presenting the first full fp4 training scheme on a 7B model. This is presented in the experiment section.

   Guidelines:

   - The answer NA 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 NA 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: In Section 6 we present the limitations of the paper.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: The theoretical results presented in Appendix B , includes all assumptions taken and is a complete proof.

   Guidelines:

   - The answer NA 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

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: The paper use open source dataset and publish the full code to reproduce all experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 includes all details about the experiments in the paper.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: We explain in the experiments section about the use of standard opensource train-test split, initialization.

   Guidelines:

   - The answer NA 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: The paper explain in Section 5 all details about the computer resources of the experiments.

   Guidelines:

   - The answer NA 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: The paper conform the Code of Ethics.

   Guidelines:

   - The answer NA 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: [Yes]

    Justification: In Appendix A we discuss the broader impacts of the paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA 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: All data used in the paper is open source. The paper include full citation of all related works.

    Guidelines:

    - The answer NA 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: [Yes]

    Justification: The paper include anonymous github with all the code to reproduce the experiments. The code include full documentation required.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLM was used only for editing or formatting purposes.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

A Broader impacts

Accelerating the runtime of large language models (LLMs) plays a pivotal role in shaping modern digital experiences, especially as systems like ChatGPT and Gemini become increasingly embedded in everyday applications. Improving their speed and efficiency addresses a major bottleneck in large-scale AI deployment. In addition to boosting performance and reducing memory demands, faster models broaden accessibility, enabling more users to adapt and innovate with LLMs for their specific needs. However, this increased accessibility also raises concerns about potential misuse, underscoring the need for responsible development and oversight.

B Analysis of quantized SGD

B.1 Analysis of quantized SGD with stochastic rounding

We study how quantization noise affects the expected loss decrease during gradient descent.

Let $L(\theta)$ be a twice-differentiable scalar loss function on $\mathbb{R}^d$.

Step 1: Start with the Taylor expansion of the loss. We consider a small step $\Delta\theta$ from θ_t. The second-order Taylor expansion of L at θ_t is:

$$L(\theta_t + \Delta\theta) = L(\theta_t) + \nabla L(\theta_t)^T \Delta\theta + \frac{1}{2}\Delta\theta^T H(\theta_t)\Delta\theta + \cdots$$

where $H(\theta_t) = \nabla^2 L(\theta_t)$.

Step 2: Apply a quantized gradient update. We use a noisy estimate of the gradient due to quantization:

$$g_q = \nabla L(\theta_t) + \varepsilon,$$

where ε is quantization noise. The update rule becomes:

$$\theta_{t+1} = \theta_t - \eta\, g_q \quad \Rightarrow \quad \Delta\theta = -\eta\, g_q.$$

Step 3: Substitute into the Taylor expansion. Plugging $\Delta\theta = -\eta\, g_q$ gives

$$L(\theta_{t+1}) \approx L(\theta_t) - \eta\, \nabla L(\theta_t)^T g_q + \frac{1}{2}\eta^2\, g_q^T H(\theta_t)\, g_q.$$

Step 4: Expectation over quantization noise. Under stochastic rounding, the noise ε satisfies

$$\mathbb{E}[\varepsilon] = 0, \qquad \mathbb{E}[\varepsilon\varepsilon^T] = \sigma_q^2 I.$$

Taking expectations in the Taylor expansion gives

$$\mathbb{E}[L(\theta_{t+1})] \approx L(\theta_t) - \eta\, \nabla L(\theta_t)^T\, \mathbb{E}[g_q] + \tfrac{1}{2}\eta^2\, \mathbb{E}\big[g_q^T H(\theta_t)\, g_q\big].$$

Since

$$\mathbb{E}[g_q] = \mathbb{E}[\nabla L(\theta_t) + \varepsilon] = \nabla L(\theta_t),$$

the linear term simplifies to

$$-\eta\, \nabla L(\theta_t)^T \nabla L(\theta_t) = -\eta\, \|\nabla L(\theta_t)\|^2.$$

Next,

$$\mathbb{E}[g_q g_q^T] = \mathbb{E}\big[(\nabla L + \varepsilon)(\nabla L + \varepsilon)^T\big] = \nabla L\, \nabla L^T + \sigma_q^2 I,$$

so

$$\mathbb{E}[g_q^T H(\theta_t)\, g_q] = \operatorname{tr}\big(H(\theta_t)\, \mathbb{E}[g_q g_q^T]\big) = \nabla L(\theta_t)^T H(\theta_t)\, \nabla L(\theta_t) + \sigma_q^2 \operatorname{tr}\big(H(\theta_t)\big).$$

Putting everything together,

$$\mathbb{E}[L(\theta_{t+1})] = L(\theta_t) - \eta\, \|\nabla L(\theta_t)\|^2 + \tfrac{1}{2}\eta^2 \Big(\nabla L(\theta_t)^T H(\theta_t)\, \nabla L(\theta_t) + \sigma_q^2 \operatorname{tr}\big(H(\theta_t)\big)\Big).$$

Step 5: Convergence dynamics with SR From Step 4, the expected change in loss is:

$$\mathbb{E}[L(\theta_{t+1}) - L(\theta_t)] \approx -\eta \|\nabla L(\theta_t)\|_2^2 + \tfrac{1}{2}\eta^2 \left(\nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t) + \sigma_q^2 \mathrm{tr}(H(\theta_t))\right) .$$

This can be written as:

$$\mathbb{E}[L(\theta_{t+1}) - L(\theta_t)] \approx \underbrace{-\left(\eta \|\nabla L(\theta_t)\|_2^2 - \tfrac{1}{2}\eta^2 \nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t)\right)}_{\text{useful descent component}} + \underbrace{\tfrac{1}{2}\eta^2 \sigma_q^2 \mathrm{tr}(H(\theta_t))}_{\text{quantization noise effect}} .$$

The useful descent component is negative if:

$$\eta \|\nabla L(\theta_t)\|_2^2 > \tfrac{1}{2}\eta^2 \nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t) \Rightarrow \eta < \frac{2\|\nabla L(\theta_t)\|_2^2}{\nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t)} .$$

A more conservative condition is:

$$\eta < \frac{2}{\lambda_{\max}(H(\theta_t))} .$$

Step 6: Optimal step size η^*. To find the optimal step size, we define

$$U(\eta) = \mathbb{E}[L(\theta_{t+1}) - L(\theta_t)] = -\eta \|\nabla L(\theta_t)\|_2^2 + \tfrac{1}{2}\eta^2 \left(\nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t) + \sigma_q^2 \mathrm{tr}(H(\theta_t))\right) .$$

Setting its derivative to 0:

$$\frac{dU}{d\eta} = -\|\nabla L(\theta_t)\|_2^2 + \eta \left(\nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t) + \sigma_q^2 \mathrm{tr}(H(\theta_t))\right) = 0 .$$

Solving for η^*:

$$\eta^* = \frac{\|\nabla L(\theta_t)\|_2^2}{\nabla L(\theta_t)^T H(\theta_t) \nabla L(\theta_t) + \sigma_q^2 \mathrm{tr}(H(\theta_t))} .$$

Step 7: Training with optimal step size η^*. Substitute $\eta^* = \frac{A}{B}$ back into

$$U(\eta) = -\eta\, A + \tfrac{1}{2}\,\eta^2\, B,$$

where

$$A = \|\nabla L(\theta_t)\|_2^2, \qquad B = \nabla L(\theta_t)^T H(\theta_t)\, \nabla L(\theta_t) + \sigma_q^2\, \mathrm{tr}\big(H(\theta_t)\big).$$

Then

$$U(\eta^*) = -\frac{A}{B}\, A + \tfrac{1}{2}\Big(\frac{A}{B}\Big)^2 B = -\frac{A^2}{B} + \frac{1}{2}\,\frac{A^2}{B} = -\frac{1}{2}\,\frac{A^2}{B},$$

i.e.

$$U(\eta^*) = -\frac{\|\nabla L(\theta_t)\|_2^4}{2\Big(\nabla L(\theta_t)^T H(\theta_t)\, \nabla L(\theta_t) + \sigma_q^2\, \mathrm{tr}(H(\theta_t))\Big)}.$$

Let $X = \nabla L^T H \nabla L$, $Y = \mathrm{tr}(H)$ and $Z = \|\nabla L\|_2^4$. Then

$$U(\eta^*) = -\frac{Z}{2\big(X + \sigma_q^2 Y\big)}.$$

Step 8: Maximum sensitivity to noise. We start with the sensitivity

$$f(\sigma_q) \;=\; \frac{\partial U(\eta^*)}{\partial \sigma_q} \;=\; \frac{Z\, Y\, \sigma_q}{\big(X + Y\, \sigma_q^2\big)^2}.$$

To find its maximum, compute the derivative w.r.t. σ_q:

$$\frac{df}{d\sigma_q} = Z\, Y\, \frac{(X + Y\sigma_q^2)^2 \;-\; \sigma_q \cdot 2\,(X + Y\sigma_q^2) \cdot (2Y\sigma_q)}{(X + Y\sigma_q^2)^4} .$$

We factor this expression and simplify it:

$$\frac{df}{d\sigma_q} = \frac{Z\, Y\, (X + Y\sigma_q^2)\big[(X + Y\sigma_q^2) - 4Y\sigma_q^2\big]}{(X + Y\sigma_q^2)^4} = \frac{Z\, Y\, [\, X - 3Y\sigma_q^2\,]}{(X + Y\sigma_q^2)^3}.$$

Set this to zero to locate the minimum:

$$X - 3\,Y\,\sigma_q^2 = 0 \quad \Longrightarrow \quad \sigma_q^2 = \frac{X}{3Y}.$$

Thus the critical noise level is

$$\sigma^2_{\text{critical}} = \frac{X}{3Y}.$$

That is:

$$\sigma^2_{\text{critical}} = \frac{\nabla L(\theta_t)^T H(\theta_t)\,\nabla L(\theta_t)}{3\,\text{tr}\big(H(\theta_t)\big)}\,.$$

Finally, assuming[3]

$$\frac{\nabla L(\theta_t)^T H(\theta_t)\,\nabla L(\theta_t)}{\|\nabla L(\theta_t)\|_2^2} \approx \frac{\text{tr}\big(H(\theta_t)\big)}{d}\,,$$

we get

$$\|\nabla L(\theta_t)\|_2^2 \approx 3\,d\,\sigma^2_{\text{critical}} \quad \Longrightarrow \quad \|\nabla L(\theta_t)\|_2 \approx \sqrt{3d}\,\sigma_{\text{critical}}.$$

Therefore,

$$\sigma_{\text{critical}} \;=\; \frac{\|\nabla L(\theta_t)\|_2}{\sqrt{3d}}\,.$$

In other words, once the average per-coordinate gradient falls to $\sqrt{3}$ times the quantization-noise std, FP4 gradients lose efficacy and it is time to switch to higher precision. As shown in Figure 4, setting the noise std to $k \cdot \sigma_{\text{critical}}$ with $k = 2.0, 1.0, 0.5$ yields markedly different convergence behaviors around that threshold. In Appendix B.2 we show a similar analysis without SR, which shows that the "useful descent" vanishes to zero as we train, while the biased-noise term remains even after long training..

B.2 Impact of nonzero mean noise (deterministic rounding)

In this section, we exemplify the problem with biased quantization schemes (such as RtN), in a simple scalar optimization problem with a quadratic loss

$$L(\theta) \;=\; \tfrac{1}{2}\,\lambda(\theta - \theta^*)^2, \;\Longrightarrow\; \nabla L(\theta) \;=\; \lambda(\theta - \theta^*)\,,$$

and a step size update with quantization noise ε whose mean is $\mu_\varepsilon = \mathbb{E}[\varepsilon] \neq 0$:

$$\theta_{t+1} = \theta_t - \eta\big(\nabla L(\theta_t) + \varepsilon\big) \;\Longrightarrow\; \mathbb{E}[\theta_{t+1}] = \mathbb{E}[\theta_t] - \eta\big(\lambda(\mathbb{E}[\theta_t] - \theta^*) + \mu_\varepsilon\big).$$

Define the error

$$e_t \;\triangleq\; \mathbb{E}[\theta_t] - \theta^*.$$

Then

$$e_{t+1} = \mathbb{E}[\theta_{t+1}] - \theta^* = \big[\mathbb{E}[\theta_t] - \eta(\lambda e_t + \mu_\varepsilon)\big] - \theta^* = e_t - \eta\lambda e_t - \eta\mu_\varepsilon\,.$$

Therefore,

$$e_{t+1} = (1 - \eta\lambda)\,e_t \;-\; \eta\,\mu_\varepsilon.$$

Unrolling this recursion gives, for $a = 1 - \eta\lambda$,

$$e_n = a^n e_0 \;-\; \eta\,\mu_\varepsilon \sum_{k=0}^{n-1} a^k.$$

Since $\sum_{k=0}^{n-1} a^k = \frac{1-a^n}{1-a}$ and $1 - a = \eta\lambda$, we get

$$e_n = a^n e_0 \;-\; \frac{\eta\,\mu_\varepsilon}{\eta\lambda}\,\big(1 - a^n\big) = a^n e_0 \;-\; \frac{\mu_\varepsilon}{\lambda}\,\big(1 - a^n\big).$$

[3] In the high-dimensional regime ($d, N \to \infty$ and $d/N \to \lambda$, where N is number of training samples), previous works [14] showed, using random-matrix theory and empirical results, that the bulk of the Hessian values can be approximately represented by the Marchenko–Pastur distribution, especially for small loss values. Thus, when $\lambda \ll 1$ (which is the common regime for LLMs), this Marchenko–Pastur distribution implies that the bulk of Hessian eigenvalues concentrates near their mean $\text{tr}(H)/d$. Empirical investigations [16, 10] confirm that the gradient predominantly occupies this bulk subspace rather than aligning with the few extreme modes. Hence the curvature experienced in the gradient direction approximates the average eigenvalue.

The loss at step n is

$$L_n = L(\,\mathbb{E}[\theta_n]\,) = \tfrac{1}{2}\,\lambda\, e_n^2 = \frac{\lambda}{2}\Big(a^n e_0 - \tfrac{\mu_\varepsilon}{\lambda}(1-a^n)\Big)^2.$$

As $n \to \infty$, $a^n \to 0$ (for $a < 0$, which is required for successful optimization), yielding the stationary error and residual loss

$$e_\infty = -\frac{\mu_\varepsilon}{\lambda}, \qquad L_\infty = \frac{\mu_\varepsilon^2}{2\lambda}.$$

Thus, instead of converging to θ^* with zero loss, biased SGD settles at

$$\mathbb{E}[\theta_\infty] = \theta^* \; - \; \frac{\mu_\varepsilon}{\lambda},$$

and leaves a residual loss

$$L(\,\mathbb{E}[\theta_\infty]\,) = \frac{\mu_\varepsilon^2}{2\lambda}.$$

C Additional Experimental Results

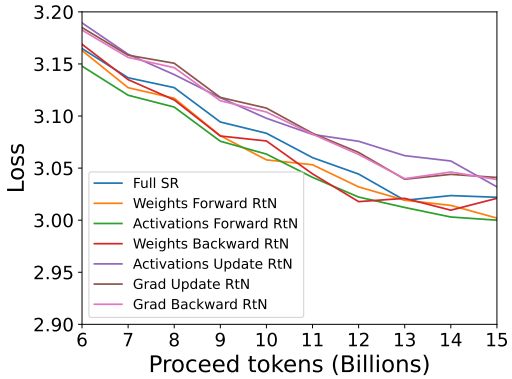


Figure 7: **Comparison of different rounding schemes when training a 350M Llama model using NVFP4 format.** In each graph, we apply RtN in one of the six elements in one of the GEMMs while the rest use SR. Notice that applying RtN to neural gradients during both the 'Update' and 'Backward' GEMMs, and to the activations during the 'Update' GEMM leads to higher training loss, while applying RtN to the other components has the opposite effect, reducing the loss.

Table 4: Training loss of Llama 125M over 30B tokens with different seeds, yield a similar loss, resulting in a standard deviation of 0.001.

Training loss	Seed
3.03	1337
3.027	1234
3.025	2345
3.027	3456
3.029	4567

Table 5: Ablation study to determine how many tokens are required to reach the same loss as the BF16 baseline. Notice the QAF ratio decrease for larger dataset. In all QAF experiments we use the peak learning rate equal to the last learning rate in the FP4 training.

Starting point	QAF length	Ratio
200B	20B	10%
500B	28B	5.6%
1T	40B	4%