MXNorm: Reusing block scales for efficient tensor normalisation

Callum McLean[*]
Graphcore
callumm@graphcore.ai

Luke Y. Prince[*]
Graphcore
lukep@graphcore.ai

Alexandre Payot
Graphcore
alexandrep@graphcore.ai

Paul Balança
Graphcore
paulb@graphcore.ai

Carlo Luschi
Graphcore
carlo@graphcore.ai

Abstract

The matrix multiplications which comprise the bulk of computation in deep learning are being performed in increasingly narrow-precision formats. For example, next generation AI accelerators support dot products in MXFP4, a format requiring only 4.25 bits per element. However, accelerator performance for low-precision matrix multiplication far outstrips accelerator performance on reductions and elementwise computations that are still being performed in higher precision. In this work, we reduce the cost of the RMSNorm layer by fusing approximating the RMS of a tensor with the computation of the MX block scales, thereby enabling a 32x decrease in the size of reductions needed for normalisation. We validate our approximation method on pre-training of Llama 3 models of 250M and 1B parameters, finding minimal loss of training accuracy compared to a baseline using RMSNorm with MXFP8 matmuls.

1 Introduction

Microscaling formats (henceforth referred to as "MX formats") were proposed by [1] as a way to quantise tensors to very few bits per element while preserving the range of higher precision formats such as BF16 and FP32. MX quantisation chunks a tensor into contiguous blocks of a fixed size and computes a scale factor for each block, which is used to rescale the elements of the block to the range of a low precision format. An MX Tensor can therefore be thought of as a tuple of an MX scale tensor comprised of block scales and an MX values tensor of rescaled, quantised elements. Using the E8M0 format (which only represents integer powers of 2) for the scales preserves the range of BF16 while adding the minimal number of bits to the representation. The scale for each block is thus chosen to be the block's absmax rounded to a power of 2 [1]. The implementation of this rounding is not fully standardised and several schemes have been proposed [2–4].

Normalisation layers are essential for ensuring pre-training stability. Various normalisation schemes have been favoured over the past decade, such as BatchNorm [5] for convolutional neural networks, LayerNorm for sequence models [6], and more recently RMSNorm [7] for large language models such as the Llama series [8–10]. In the case of RMSNorm, each token's hidden state is normalised using its root mean square. Placement of norms is also key to pre-training performance, with frontier models typically placing norms at the start of each residual branch [11].

We make two observations: (1) MX quantisation and RMSNorm both gather statistics on the tensor to rescale elements (although the former is scale-preserving and the latter is scale-rectifying), and (2)

[*]Equal Contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ML for Systems.

when a probability distribution is scaled linearly, the expected absmax of the distribution is scaled accordingly. From these observations we propose to approximate the RMS using the block scales calculated during MX quantisation, thereby enabling us to fuse RMSNorm with MX quantisation for activations and requiring only a single pass of statistics gathering over the whole tensor. Given the stark difference in FLOPs ($10 - 100\times$) between elementwise/reduction (non-matmul) ops and matmul ops in modern hardware [12, 13], there is now plenty to gain by minimising non-matmul FLOPS when designing compute blocks. We call this new scheme **MXNorm** and demonstrate its effectiveness in pre-training of language models of up to 1B parameters.

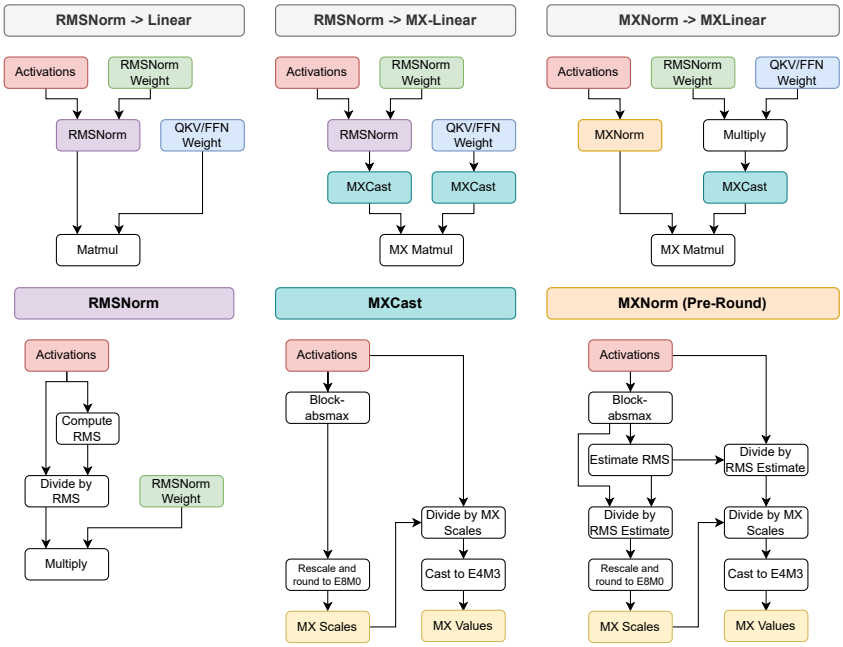

Figure 1: Computational graphs for RMSNorm, MXCast, and MXNorm in the context of Norm + Linear layer pattern. Top left: RMSNorm + Linear graph for high precision training. Top middle: RMSNorm + Linear graph with linear inputs cast to MX (MXLinear). Top right: RMSNorm approximated with MXNorm and RMSNorm weight fused with Linear weight. Bottom left: RMSNorm graph. Bottom middle: MXCast graph for MXFP8 activations. Note that MXCast is applied to weights (E4M3 values) and gradients (E5M2 or E4M3 values) as well. Bottom right: MXNorm graph for pre-round scheme.

2 Methods

Here we will briefly describe RMSNorm, MXFP casting, and our main contribution: MXNorm. The differences between each of these schemes and their use in a Norm + Linear layer is summarised in Figure 1.

2.1 RMSNorm

Given an activation tensor $X \in \mathbb{R}^{N \times D}$, RMSNorm normalises each token of the tensor X using the calculated RMS S, given by:

$$S_i = \sqrt{\frac{1}{D} \sum_{k=1}^{D} X_{ik}^2} \tag{1}$$

The learnable gain parameter $\gamma \in \mathbb{R}^D$ rescales the normalised X along the hidden dimension to give output Z as follows:

$$Z_{ij} = (X_{ij}/S_i)\gamma_j \tag{2}$$

In the Llama 3 architecture, there is an RMSNorm layer immediately prior to the QKV projection in each attention layer and immediately prior to the input and gate projections of the FFN (which can be fused into a single matmul), following the pre-norm architecture [11]. This leads to the "RMSNorm -> Linear" pattern shown in the top-left of Figure 1.

2.2 Conversion to MX

For a block $B \in \mathbb{R}^K$ with K entries, the scale value $\widetilde{B}^{(s)}$ is given by

$$\widetilde{B}^{(s)} := \texttt{pow2\_round}(\texttt{rescale}(\max_k |B_k|)) \tag{3}$$

where the function `pow2_round` rounds its input to a power of 2, and `rescale` divides the input by the largest power of 2 representable by the MX-values data format, e.g., 256 for E4M3. The details of `pow2_round` are implementation defined, with possible options given by [1–3]. For our experiments we use the method defined in [2], enumerated as `ScaleCalculationMode.RCEIL` in TorchAO [4].

2.3 Approximation of RMS during MX quantisation

We observe that the RMS of a zero-centred distribution correlates strongly with the expected maximum absolute value of a block of input samples (Figure 2, left). Motivated by this, we define two possible approximations to the RMS using these block absmaxes either before or after `pow2_round` is applied.

There is a linear relationship between the mean block absmax and the expected RMS of the data distribution (Figure 2, left). We empirically estimate the expected ratio c between the RMS and the mean of the block absmaxes. We use this coefficient to rescale the block-absmax before rounding to MX-scales in what we call the *pre-round* scheme. That is, if $D = MK$ where K is the block size, we have $\tilde{S}_i = c\,\frac{1}{M}\sum_m \max_k |X_{imk}|$, where X is reshaped to have shape (N, M, K) for simplicity.

In contrast, the relationship between the mean rounded block-absmax and RMS of the data distribution is not linear and indeed has no tractable form (see Figure 2, left). Instead, we model this monotonic function with a piecewise linear approximation derived empirically from a Gaussian assumption, exploiting the fact that the function is cyclic on a logarithmic scale to make the number of pieces finite (see Appendix F for full details). This gives us our *post-round* scheme.

We demonstrate the fidelity of our approximation by comparing the output of a modified `mx_quantise` function `mx_norm` on the distribution of scales and value tensors on RMSNorm followed by `mx_quantise`. The middle two panels of Figure 2 shows that the distribution of scales and values using the pre-round scheme is almost identical whereas the post-round scheme represents larger scales with higher frequency, which in turn slightly decreases the mode of the distribution of values. We also demonstrate that the approximation quality as measured by r^2 improves asymptotically towards 1 as the number of blocks increases (see Figure 2, right).

2.4 MXNormLinear

To build an MXNormLinear layer, we take as our starting point an RMSNorm layer followed by a Linear layer (ignoring biases). This takes input X, linear layer weights W, and an affine norm gain parameter γ and outputs $Y = ZW^\top$, where Z is defined as in Equations 1 and 2.

For an MXNormLinear layer we approximate the RMS S_i using one of the approaches described in Section 2.3 to produce $\tilde{S}_i$. Since the output of MXNorm must be an MX Tensor and the inputs to the MX-matmul must also be an MX Tensor, we cannot apply the norm gain and so cannot materialise Z directly. Instead we apply these affine gains to the weights to produce the fused weight W' where $W'_{ij} = W_{ij}\gamma_j$ and compute

$$Y = \texttt{mx\_norm}(X)\,\texttt{mx\_quantise}(W')^\top \tag{4}$$

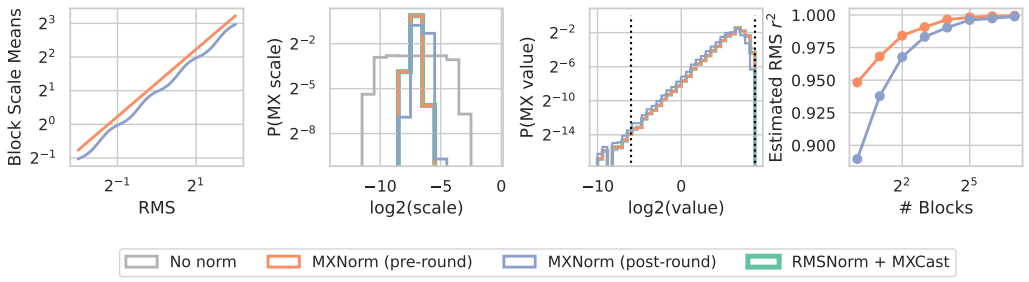

Figure 2: MXNorm as an approximation of RMSNorm. Left: Relationship between the mean of block scales and the RMS. (Both axes are on a log scale.) Middle Left: MX scale distribution of normalised tensors. Middle Right: MX value distribution of normalised tensors. Right: MXNorm r^2 goodness-of-fit approaches 1 with more blocks.

where the output Y is accumulated in higher precision. The details of the gradient of MXNormLinear are given in Appendix C. A PyTorch implementation of the MXNormLinear forward and backward pass is given in Appendix D.

3 Experiments and Results

Figure 3: Learning rate sensitivity of MXNorm compared to RMSNorm. Left: 250M parameter model (depth=4, width=2048). Right: 1B parameter model (depth=16, width=2048)

.

We validate MXNorm on pretraining of Llama 3 models [10] of different sizes on the SlimPajama dataset [14], comparing against a baseline of RMSNorm followed by MXLinear layers. For full details, please refer to Appendix A.

We examined pre-training stability by running a learning rate sweep on 250M parameter and 1B parameter models trained on 5B and 20B tokens respectively. The effect of quantisation is often felt on training stability and can be seen at smaller scales by examining learning rate sensitivity [15].

In Figure 3, we demonstrate that there is a small degradation in the training loss ([RMS + MXLinear] 250M: 3.14, 1B: 2.63) for MXNorm schemes that is slightly smaller for the pre-round scheme (250M: 3.13, 1B: 2.67) vs. the post-round scheme (250M: 3.16, 1B: 2.68). In addition, as learning rate increases the pre-round scheme maintains a loss closer to the baseline than the post-round scheme, indicating greater training stability. This trend can be seen at both 250M and 1B parameter scales. We show loss curves for the optimum learning rate of 1B models in Appendix E. The presence of loss spikes with MXNorm schemes further indicates a slight loss in training stability.

4 Conclusion

We demonstrate the possibility of estimating the RMS during calculation of MX-scales with minimal overhead in a way that removes the need for RMSNorm in LLM pre-training. We show at up to 1B scale that this approach leads to minimal loss of pre-training performance.

It remains to be seen whether MXNorm is sufficiently stable for larger scale pre-training and whether the benefits can be realised as wall-clock speedups in pre-training and inference. Future work could also consider methods to convert pre-trained models using RMSNorm to using MXNorm using a method like post-training quantisation.

References

[1] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, 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 Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning, 2023. URL https://arxiv.org/abs/2310.10537.

[2] Asit Mishra, Dusan Stosic, and Simon Layton. Recipes for pre-training llms with mxfp8, 2025. URL https://arxiv.org/abs/2506.08027.

[3] PyTorch AO Team. Scalecalculationmode.even. GitHub repository, 2024. URL https://github.com/pytorch/ao/blame/e6b38bb0e1477ae6aaca0a3d30de70598be43290/torchao/prototype/mx_formats/config.py#L69-L72. Lines 69-72 in torchao/prototype/mx_formats/config.py, commit e6b38bb.

[4] Andrew Or, Apurva Jain, Daniel Vega-Myhre, Jesse Cai, Charles David Hernandez, Zhenrui Zheng, Driss Guessous, Vasiliy Kuznetsov, Christian Puhrsch, Mark Saroufim, Supriya Rao, Thien Tran, and Aleksandar Samardžić. Torchao: Pytorch-native training-to-serving model optimization, 2025. URL https://arxiv.org/abs/2507.16099.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[7] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[9] Llama Team. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[10] Llama Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[11] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020. URL https://arxiv.org/abs/2002.04745.

[12] Jacob Austin, Sholto Douglas, Roy Frostig, Anselm Levskaya, Charlie Chen, Sharad Vikram, Federico Lebron, Peter Choy, Vinay Ramasesh, Albert Webson, and Reiner Pope. How to scale your model. 2025. URL https://jax-ml.github.io/scaling-book/tpus/. Retrieved from https://jax-ml.github.io/scaling-book/.

[13] NVIDIA Corporation. Nvidia blackwell architecture technical overview. Technical overview, NVIDIA Corporation, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture. Accessed: 2025-08-19.

[14] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

[15] Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities, 2023. URL https://arxiv.org/abs/2309.14322.

[16] Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. Torchtitan: One-stop pytorch native solution for production ready llm pre-training, 2025. URL https://arxiv.org/abs/2410.06511.

Contents

A Experimental Details

For all experiments we use the TorchTitan [16] distributed pre-training library with FSDP in conjunction with TorchAO [4] for MX quantisation.

We use the hyperparameters in Table 1 for our experiments. For learning rate sweeps we increment in powers of 2 from 2^{-14} to 2^{-3}. In all other cases we reuse the default hyperparameters used for Llama 3 pre-training in TorchTitan using FSDP and BFloat16.

Table 1: Pretraining configuration for Llama 3 models

	Model Size	
	250M	1B
Global batch size	32	256
Sequence length	4096	4096
Total training tokens	5.24B	20.97B
Transformer layer Count	4	16
Hidden dimension	2048	2048
Q:KV head ratio	4:1	4:1
FFN dimension	3072	3072

Each transformer layer contains both an attention layer and a feedforward network. Note that our 250M model preserves the width of the 1B model, only reducing layer count. We chose this method of scaling down models to preserve the accuracy of our approximation to $r^2 > 0.99$ for both pre-rounded and post-rounded estimates of the RMS via MXNorm (see Figure 2).

B Compute resources

Our 250M models are trained in 90 minutes on a single node with 8 NVIDIA H100s connected by NVLink. Our 1B models are trained in 6 hours on 4 nodes comprising 32 NVIDIA H100s connected by NVLink. We thank LambdaLabs for providing the compute for this project.

C Gradient calculation of MXNormLinear

For the gradient calculation we approximate the gradient of RMSNorm followed by a linear layer. Given the gradient of the loss with respect to the output ∇Y, the backwards pass for RMSNorm is given by:

$$(\nabla Z) = (\nabla Y)W \tag{5}$$

$$(\nabla W) = (\nabla Y)^\top Z \tag{6}$$

$$(\nabla \gamma)_j = \sum_k (\bar{X}_{kj} \cdot (\nabla Z)_{kj}) \tag{7}$$

$$(\nabla \bar{X}) = (\nabla Z)\gamma \tag{8}$$

$$U_{ij} = \left(\sum_k (\nabla \bar{X})_{ik} X_{ik} \right) X_{ij} \tag{9}$$

$$(\nabla X) = (S^{-1})^\top \nabla \bar{X} - \frac{1}{D}(S^{-3})^\top U \tag{10}$$

In the above, we introduce the term $\bar{X}$ defined by $\bar{X}_{ij} := X_{ij}/S_i$. In addition, we define $S^{-1} \in \mathbb{R}^N$ to be given by $(S^{-1})_i = S_i^{-1}$ and similarly for S^{-3}. For the gradient of MXNormLinear, we reuse the calculation of RMS-Norm backward as a straight-through estimator of MXNorm, using a cached RMS estimate $\tilde{S}$ described in Section 2.3.

We must take care when quantising Z to MX format since we must quantise along the columns of Z rather than the rows as in the forward pass. We again re-use the cached RMS estimate to materialise a high-precision form of $X/\tilde{S}$ before quantising. We also defer applying the affine norm parameters for the gradient of weights ∇W until after the MX-matmul i.e.,

$$(\nabla W) = \texttt{mx\_quantise}(\nabla Y)^\top \texttt{mx\_quantise}(X/\tilde{S}) \cdot \gamma \tag{11}$$

D Implementation of MXNormLinear

We provide a PyTorch implementation for the forward and backward pass of MXNormLinear using the pre-norm scheme. We omit the details of `pow2_round` since there is no standardised implementation [4]. In our experiments we use the implementation defined by [2].

```
import torch
from torchao.prototype.mx_formats.mx_tensor import MXTensor
from math import log2, floor
from mx_norm_utils import pow2_round

def absmax_scale_factor(block_size):
    """
    Estimated RMS(X)/E[max(abs(X))] using monte carlo sampling
    from Gaussian distribution
    """
    if block_size == 16:
        return 0.4817
    elif block_size == 32:
        return 0.4260
    elif block_size == 64:
        return 0.3850

def get_largest_pow2(dtype):
    return 2 ** floor(log2(torch.finfo(dtype).max))

def mx_quantise(x, mx_data_dtype, block_size):
    x_blocked = x.reshape(*x.shape[:-1], -1, block_size)
    block_absmax = x_blocked.abs().amax(dim=-1, keepdim=True)
    largest_pow2 = get_largest_pow2(mx_data_dtype)
    mx_scales = pow2_round(block_absmax / largest_pow2)
    mx_data = x_blocked / mx_scales.unsqueeze(-1)
    mx_data = mx_data.reshape(x.shape)
    mx_data = mx_data.to(mx_data_dtype)
    mx_scales = mx_scales.to(torch.float8_e8m0fnu)
    return MXTensor(mx_scales, mx_data, mx_data_dtype, block_size, x.dtype)

def mx_norm(x, mx_data_dtype, block_size):
    x_blocked = x.reshape(*x.shape[:-1], -1, block_size)
    block_absmax = x_blocked.abs().amax(dim=-1, keepdim=True)

    # absmax_scale_factor returns the expected ratio
    # of the RMS divided by the mean of the absmaxes
    coef = absmax_scale_factor(block_size)
    rms_estimate = block_absmax.mean(dim=-1, keepdim=True) * coef
    scaled_block_absmax = block_absmax / rms_estimate
    largest_pow2 = get_largest_pow2(mx_data_dtype)
    mx_scales = pow2_round(scaled_block_absmax / largest_pow2)

    # Creating data tensor: want mx_scale * mx_data = x / rms_estimate
    # So we want mx_data = (x / rms_estimate) / mx_scale
    # Need to cast MX scales back to match dtypes for divide
    mx_data = (x_blocked / rms_estimate) / mx_scales
    mx_data = mx_data.reshape(x.shape)
    mx_data = mx_data.to(mx_data_dtype)
```

```
    return MXTensor(mx_scales, mx_data, mx_data_dtype, block_size, x.dtype)


def mx_norm_linear_forward(x, norm_weight, linear_weight, mx_data_dtype, block_size):
    mx_normalised_activations = mx_norm(x, mx_data_dtype, block_size)
    mx_fused_weight = mx_quantise(norm_weight * linear_weight, mx_data_dtype, block_size)
    return torch.mm(mx_normalised_activations, mx_fused_weight.t())

def rms_norm_grad(grad_out, x, rms):
    delta = torch.mean(grad_out * x, dim=-1, keepdim=True)
    grad_x = rms.pow(-1) * grad_out - rms.pow(-3) * x * delta
    return grad_x

def mx_norm_linear_backward(
        grad_out, rms_estimate, x, norm_weight, linear_weight, mx_data_dtype, block_size
    ):
    # Need grad_out to be mx_quantised along both rows and columns
    mx_grad_out = mx_quantise(grad_out, mx_data_dtype, block_size)
    mx_t_grad_out = mx_quantise(grad_out.t(), mx_data_dtype, block_size)

    # Divide by rms_estimate from forward pass in higher precision
    normed_x = x / rms_estimate
    mx_t_normed_x = mx_quantise(normed_x.t(), mx_data_dtype, block_size)
    mx_linear_weight = mx_quantise(linear_weight, mx_data_dtype, block_size)

    # Compute parameter gradients
    grad_mm_input = torch.mm(mx_grad_out,  mx_linear_weight.t())
    grad_linear_weight = torch.mm(mx_t_grad_out, mx_t_normed_x.t()) * norm_weight
    grad_norm_weight = torch.sum(normed_x * grad_mm_input, dim=0)

    # Compute gradient w.r.t input
    # Use rms_norm_grad as a straight through estimator for mx_norm
    grad_normed_x = grad_mm_input * norm_weight
    grad_x = rms_norm_grad(grad_normed_x, x, rms_estimate)
    return grad_x, grad_norm_weight, grad_linear_weight
```

E Convergence of Llama 3 1B with MXNorm

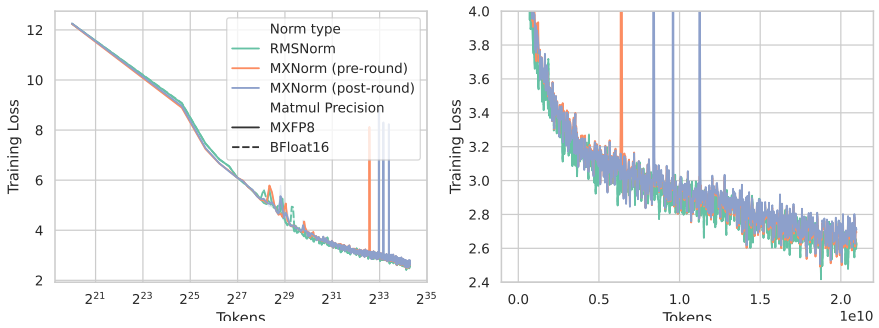

Figure A.1: Training loss convergence of 1B parameter models trained on 20B tokens with MXNorm and RMSNorm. Left: Training loss shown with x-axis on log scale to highlight early training behaviour. Right: Training loss shown with x-axis on linear scale to highlight later training behaviour.

F Post-round MXNorm

F.1 Approximating the average of the rounded scale factors

We wish to calculate the expected value of the rounded MX scale factors assuming the tensor is drawn from a Gaussian distribution with mean zero and unknown standard deviation. Suppose $X_i \sim \mathcal{N}(0, \sigma^2)$ for $0 \leq i < K$ where K is the MX block size and the X_i's are independent. We define $r(x) := 2^{\lfloor log_2(x) \rfloor}$ for the rounded MX scale factors in E8M0 (ignoring the `rescale` operation in Equation 3, which amounts to multiplication by a constant and hence can be factored out of the calculations).

We then have

$$\mathbb{E}(\max_i(r(X_i))) = \sum_{j=-\infty}^{\infty} 2^j \cdot \mathbb{P}(\max_i(r(X_i)) = 2^j)$$

We therefore need to compute $\mathbb{P}(\max_i(r(X_i)) = 2^j)$. We have

$$\mathbb{P}(\max_i(r(X_i)) = 2^j)$$

$$= \mathbb{P}(\exists i : r(X_i) = 2^j \wedge \forall i : r(X_i) \leq 2^j)$$

$$= \mathbb{P}(\exists i : 2^j \leq |X_i| < 2^{j+1} \wedge \forall i : |X_i| < 2^{j+1})$$

Applying the law of probability that $\mathbb{P}(A \wedge B) = \mathbb{P}(B) \cdot \mathbb{P}(A|B)$ gives:

$$= \mathbb{P}(\forall i : |X_i| < 2^{j+1}) \cdot \mathbb{P}\left(\exists i : 2^j \leq |X_i| < 2^{j+1} \middle| \forall i : |X_i| < 2^{j+1}\right)$$

$$= \mathbb{P}(|X_0| < 2^{j+1})^K \cdot \left(1 - \mathbb{P}\left(\forall i : |X_i| < 2^j \middle| \forall i : |X_i| < 2^{j+1}\right)\right)$$

The simplification of the left-hand term of the product comes from the fact that the X_i's are independent and identically distributed (the choice of X_0 is arbitrary).

Applying the law of probability that if $C \Rightarrow D$ we have $\mathbb{P}(C|D) = \mathbb{P}(C \cup D)/\mathbb{P}(D) = \mathbb{P}(C)/\mathbb{P}(D)$ gives:

$$= \mathbb{P}(|X_0| < 2^{j+1})^K \cdot \left(1 - \frac{\mathbb{P}(|X_0| < 2^j)^K}{\mathbb{P}(|X_0| < 2^{j+1})^N}\right)$$

$$= \mathbb{P}(|X_0| < 2^j)^K - \mathbb{P}(|X_0| < 2^{j+1})^K$$

Note that if $X_0 \sim \mathcal{N}(0, \sigma^2)$, then (using the symmetry of the Gaussian distribution):

$$\mathbb{P}(|X_0| < x) = \mathbb{P}(X_0 < x) - \mathbb{P}(X_0 \leq -x) = \mathbb{P}(X_0 < x) - (1 - \mathbb{P}(X_0 \leq x))$$
$$= \mathbb{P}(X_0 < x) - (1 - \mathbb{P}(X_0 < x)) = 2 \cdot \mathbb{P}(X_0 < x) - 1$$

Using the CDF of the standard normal distribution $\Phi(\cdot)$, we therefore have the following:

$$\mathbb{E}(\max_i(r(X_i))) = \sum_{j=-\infty}^{\infty} 2^j \cdot \left(\left(2 \cdot \Phi(2^j/\sigma) - 1\right)^K - \left(2 \cdot \Phi(2^{j+1}/\sigma) - 1\right)^K\right)$$

As $|j|$ increases, the term in the sum rapidly decreases, so the sum can be truncated from a sum over $j \in \mathbb{N}$ to a sum over $-J < j < J$ for large J with little loss in accuracy. This truncated sum can then be calculated using any mathematical software package that supports evaluating the CDF of the standard normal distribution (including PyTorch).

F.2 Approximating the RMS from the MX tensor

For a fixed block size K, we define $f : \mathbb{R}^+ \to \mathbb{R}^+$ to be given by $f(\sigma) = \mathbb{E}(\max_i(r(X_i)))$ when $X_i \sim \mathcal{N}(0, \sigma^2)$ for $0 \leq i < K$ and the X_i's are independent.

We observe that f is strictly increasing since if the standard deviation is greater we expect the rounded scale factors to be greater, and therefore f is invertible. Hence given the mean of the rounded scale factors $\bar{X}$ we can approximate the RMS of the original values as $f^{-1}(\bar{X})$.

Since f is strictly increasing, we can compute $f^{-1}(\bar{X})$ to an arbitrary degree of precision by finding σ_1, σ_2 such that $f(\sigma_1) < \bar{X} < f(\sigma_2)$ and then iteratively narrowing this range using a binary search.

However, this is computationally expensive to do at every layer in a model. We observe that $f(2\sigma) = 2f(\sigma)$ (since $r(2x) = 2r(x)$ and $\max_i |2X_i| = 2 \max_i |X_i|$). Thus we can pre-compute $f^{-1}(2^{i/A})$ for $0 \leq i \leq A$ for some A and approximate f^{-1} on the interval $[1, 2]$ using linear interpolation on these pre-computed values, and approximate $f^{-1}(x)$ elsewhere using the identity $f^{-1}(x) = 2^{-k} f^{-1}(2^k x)$, where k is chosen such that $2^k x \in [1, 2]$.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: We claim to show minimal degradation in pre-training loss at 1B parameter scale and demonstrate this in Figure 3.

   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: We acknowledge in Section 4 that MXNorm appears to be slightly less stable than RMSNorm, which needs further investigation for models larger than 1B parameters.

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: [NA]

   Justification: This paper is primarily empirical, though we do provide a derivation of how the piecewise linear approximation for the post-round scheme can be computed.

   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: Full experimental details are provided in Appendix A and a PyTorch implementation is provided in Appendix D. We acknowledge that not everyone has access to the computational resources to reproduce this work in a reasonable amount of time.

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: We use open datasets [14] and open-source libraries for training [16] and quantisation [4].

   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: We provide these in Section 3 and Appendix A.

   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: [No]

   Justification: The experiments require significant computational resources and computing error bars would increase the amount of compute needed by a significant factor.

   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: We provide details of the compute used in Appendix B.

   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: This paper focuses on efficient pre-training and open source web-scale datasets. We acknowledge Jevon's Paradox, that making systems more efficient may paradoxically lead to increased energy use.

   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: [No]

    Justification: This paper is a foundational work on efficient pre-training and as such does not have any particular societal impact.

    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: We acknowledge the use of open source datasets and libraries.

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: [NA]

Justification: We do not release any assets alongside this work.

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: This 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: This 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: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.