NORMALIZATION-FREE RESIDUAL NETWORKS BY BOUNDING WEIGHTS AND RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Normalization layers are widely used for stabilizing deep residual networks, yet they incur computational overhead and obscure training dynamics. We propose a stable, normalization-free residual block rooted in constrained nonlinear optimization that replaces traditional activation normalization with an inexpensive, element-wise clipping operation on the output of each residual branch. We extend this framework to weights, combining it with a parameterization that enables a single global constant to bound the magnitudes of both activations and weights. This parameterization additionally results in controllable and uniform effective learning rates across all layers. Compared to networks that do not use clipping, our approach yields activation and weight distributions that are remarkably smooth and free of outliers, enabling direct post-training quantization of weights to INT4 with only a minor loss degradation. Our experiments demonstrate that this approach successfully eliminates the need for both activation and weight normalization, offering a simple and principled foundation for training deep residual networks.

1 INTRODUCTION

The training of deep residual networks is typically stabilized with normalization layers. These layers are intended to control the magnitude of activations across the depth of the network, such that activations neither explode nor vanish during training. In practice, however, normalization rescales activations and gradients by data-dependent statistics, which can vary substantially across depth and training iteration. This introduces costly reduction operations and can attenuate gradients in deeper layers relative to earlier ones. Meanwhile, training is often plagued by several sources of instability, such as activation and weight outliers, uncontrolled parameter growth, and excessively large logits. This is particularly detrimental for quantization which relies on predictable, bounded ranges for activations and model weights.

This work argues that stable training does not fundamentally require normalization; it requires a bounded range. Based on this insight, we replace normalization with explicit magnitude control via simple element-wise clipping of activations and weights. By enforcing strict boundaries, not only do norms remain finite, but the dynamic range of activations and weights is mathematically guaranteed. Networks with element-wise bounds instead of normalization operations can achieve similar performance while being substantially better suited for quantization.

Our contributions are threefold. First, we show that element-wise bounding is sufficient to replace normalization and achieve competitive performance. Second, to properly optimize this bounded architecture, we formulate the training process as a constrained nonlinear optimization problem. This results in a modified gradient clipping function that preserves inward-pointing gradients at the clipping boundary, actively biasing the optimization process to avoid uncontrolled activation growth rather than merely truncating outliers. Third, we extend this framework to weights, thereby ensuring bounded parameter magnitudes. This in turn yields a single global hyperparameter for bounding both activations and weights, controllable effective learning rates, and bounded dynamic ranges. Together, these ingredients provide a simple, efficient, and principled foundation for training deep normalization-free residual networks with well-controlled internal dynamics and strong post-training quantization performance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

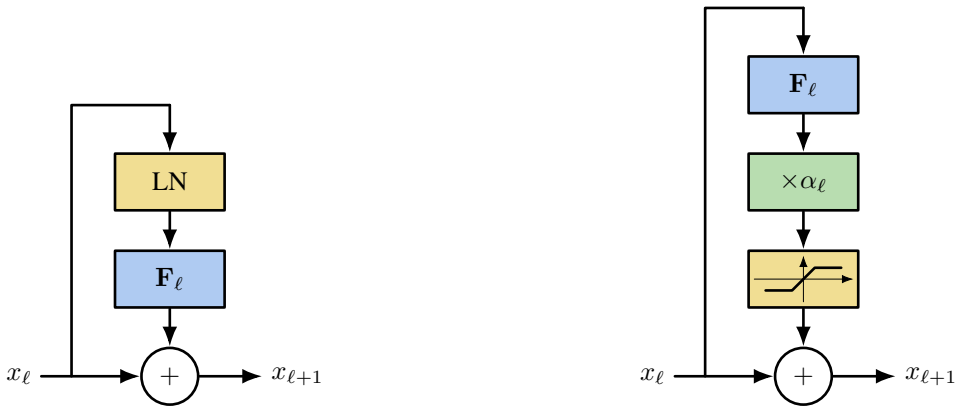
2 Related Work

Normalization layers such as BatchNorm (Ioffe & Szegedy, 2015) and LayerNorm (Ba et al., 2016) are widely used to stabilize and accelerate deep neural network training. However, normalization also introduces data-dependent reduction operations and rescales both activations and gradients by statistics that vary across depth and training (Huang et al., 2020) in residual architectures. This can obscure the underlying training dynamics and complicate efficient low-precision implementations.

A large body of work has therefore explored normalization-free training. In deep residual networks, the main difficulty is that skip connections accumulate residual updates, and activation magnitudes can grow across depth (Zhang et al., 2019; Noci et al., 2022; Dong et al., 2021; He et al., 2023). Prior approaches address this by suppressing the residual branch at initialization or by introducing explicit learnable scaling factors, as in Fixup, ReZero, LayerScale, and related methods (Zhang et al., 2019; Bachlechner et al., 2021; Touvron et al., 2021; Peebles & Xie, 2023). While effective, these methods typically rely on carefully chosen initialization or layer-dependent scaling rules. Other papers propose to replace LayerNorm by tanh-like functions (Zhu et al., 2025), which requires careful tuning of hyperparameters per model and per task.

A related line of work controls parameter scale rather than activation scale. Weight Normalization (Salimans & Kingma, 2016) and Weight Standardization (Brock et al., 2021) decouple direction and magnitude, while more recent methods such as Forced Weight Normalization (Karras et al., 2024) and nGPT (Loshchilov et al., 2025) enforce strict normalization of weights, activations, or both throughout training. These approaches improve optimization but require repeated norm computations and explicit renormalization steps.

Bounding offers a simpler alternative to normalization. Spectral normalization constrains operator norms to control Lipschitz growth (Miyato et al., 2018; Zhai et al., 2023), but is comparatively expensive. Element-wise clipping has been used in low-precision and binarized networks (Courbariaux et al., 2015; Merolla et al., 2016) and to enforce Lipschitz constraints in Wasserstein GANs (Arjovsky et al., 2017). Our work builds on this idea, but instead of using clipping merely as a practical safeguard, we cast both activation and weight bounding as a constrained nonlinear optimization problem. This yields a normalization-free residual architecture with explicit dynamic-range control, without repeated normalization operations during training or inference.



(a) Classic residual block with pre-layer norm

(b) Residual block without normalization

Figure 1: The layer normalization (LN) in (a) before computing the residual $\mathbf{F}_\ell$ can be avoided in (b) by multiplying the residual by a learnable scalar α_ℓ before clipping it to a predefined range.

3 Bounding instead of Normalization

We propose a residual block that uses element-wise clipping instead of normalization. As illustrated in Fig. 1 and in Algorithm 1, the ℓ-th residual block computes

$$\mathbf{x}_{\ell+1} := \mathbf{x}_\ell + \text{clip}(\alpha_\ell \cdot \mathbf{F}_\ell(\mathbf{x}_\ell), c), \tag{1}$$

where α_ℓ is a learnable scalar and $\mathbf{F}_\ell$ denotes a residual mapping. Residual mappings (He et al., 2016) may include attention layers, MLPs, and convolutions. The element-wise clipping function

$$\text{clip}(x, c) := \max\{-c, \min\{x, c\}\} \tag{2}$$

bounds the scaled residual by the constant $c \in \mathbb{R}^+$ and acts as the identity on inputs in $[-c, c]$. In particular, for any input $\mathbf{x}_0$ with finite norm, the output $\mathbf{x}_{\ell+1}$ also has finite norm because the residual term is bounded element-wise by construction. Hence, the norm of activations on the skip link can only grow linearly in ℓ. Additionally, outlier activations cannot affect the skip link activation $\mathbf{x}_{\ell+1}$.

The scaling coefficient α_ℓ plays a central role in controlling the contribution of the residual branch during training. We set $\alpha_\ell = 0.01$ at the start of training. Small coefficients ensure that the residual block in Eq. (1) closely approximates the identity map at initialization, so that each block contributes only a small perturbation to the main signal path while preserving comparable gradient scales across layers. This design is consistent with prior work that stabilizes deep residual networks by initializing them near the identity (Zhang et al., 2019; Bachlechner et al., 2021; Brock et al., 2021). To provide finer resolution for small values, we reparameterize α_ℓ in terms of a learnable parameter α'_ℓ:

$$\alpha_\ell := \begin{cases} \alpha'_\ell & \text{if } \alpha'_\ell \geq 1, \\ e^{\alpha'_\ell - 1} & \text{otherwise,} \end{cases} \tag{3}$$

which provides finer, logarithmic resolution for values below 1, and correspondingly initialize α'_ℓ to $1 + \ln(0.01)$. The scaling factors α_ℓ can be folded into the weights after training.

3.1 Training by Constrained Nonlinear Optimization

Training the residual block (1) with element-wise clipping (2) can be considered a constrained nonlinear optimization problem in the sense of Karush–Kuhn–Tucker (KKT) theory (Karush, 1939; Kuhn & Tucker, 1951). Let $f(\mathbf{x})$ denote the training objective as a function of a vector $\mathbf{x}$ whose elements are constrained to the interval $[-c, c]$. The feasible set can then be written in terms of the inequality constraints

$$g_{2h}(\mathbf{x}) := x_h - c \leq 0 \qquad \text{and} \qquad g_{2h+1}(\mathbf{x}) := -x_h - c \leq 0. \tag{4}$$

Each element x_h is therefore subject to an upper and a lower bound, and at most one of the two associated constraints can be active. The corresponding Lagrangian is $\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) := f(\mathbf{x}) + \sum_i \mu_i g_i(\mathbf{x})$, with multipliers $\mu_i \geq 0$. By complementary slackness, inactive constraints have $\mu_i = 0$, whereas active constraints on the boundary may contribute nonzero multipliers. At such boundary points, the gradient of the objective is balanced by the gradients of the active constraints by setting $\nabla_{\mathbf{x}} \mathcal{L} = 0$, yielding the usual projected-gradient interpretation for box-constrained optimization (Nocedal & Wright, 2006). In other words, if a gradient step moves $\mathbf{x}$ outside the feasible set, the update will be projected back onto the admissible domain.

Under standard backpropagation through the clipping function (2), the derivative is one for $|x_h| < c$ and zero otherwise. Consequently, the gradient propagates only in the interior of the interval. However, in constrained nonlinear optimization the inward-pointing gradients remain active at the boundary as they move the parameters back into the feasible region. We therefore modify the per-sample mini-batch gradients $\boldsymbol{\gamma} := \nabla_{\mathbf{x}} f(\mathbf{x})$ before aggregation by the optimizer (see Algorithm 1) as

$$\gamma'_h := \begin{cases} \gamma_h, & \text{if } |x_h| < c \text{ or } x_h \cdot \gamma_h \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Since the gradient descent updates in the direction $-\gamma$, the inward-pointing gradient is retained for $x_h \cdot \gamma_h \geq 0$. The gradient is suppressed only when the descent step would move farther outward. In practice, we compare $|x_h|$ with the slightly smaller threshold $c - \epsilon$ to make boundary detection robust in floating-point arithmetic. As a result, clipping not only enforces the bound in the forward pass, but also encourages the parameters to satisfy the constraint already before the clipping operation. In our residual blocks (1), we use this clipping operation for both activation and weight bounding (see Section 3.2). While element-wise gradient projection (5) trivially solves weight bounding in the standard KKT formulation (4), its accuracy for the nested activation clipping operations (1) will depend on the training setup and resulting chained derivatives, see Appendix G. We show in Appendix F.4 that this modification successfully constrains activations compared to standard clipping (Fig. 10).

A smooth alternative such as the hyperbolic tangent may appear attractive because it avoids hard discontinuities, but it enters a saturation regime for large inputs. In that regime, the derivative approaches zero, which weakens gradient flow and can slow optimization. In the ablations we show that tanh weight clamping results in much higher loss (see Appendix F.5). Clipping is therefore particularly preferable for weights in our setting, allowing uniformly linear updates on $[-c, c]$, while exactly enforcing strict boundaries outside that interval. For activation clipping, the advantage over tanh clamping is minor (see Appendix F.4). We find robust bounding of both weights and activations is crucial for robust training (see Fig. 5), whereas the approach of Zhu et al. (2025) to only clamp activations using tanh at the input of residual blocks was not sufficient in the general case.

3.2 Weight Bounding and Learning Rate Control

Rather than controlling the effective learning rates of the weights through weight normalization (Karras et al., 2024; Loshchilov et al., 2025), we control them by bounding the weights within the same constrained nonlinear optimization framework of Eq. (5). This naturally complements activation bounding by limiting uncontrolled activation growth, which would otherwise require aggressive clipping. Following Karras et al. (2024), we parameterize each weight matrix during training as

$$W := \frac{1}{\sqrt{d_{\text{in}}}} \cdot \widetilde{W}, \tag{6}$$

where d_{in} is the input dimension and the underlying parameters $\widetilde{W}$ are initialized from a standard normal distribution. Writing $s := \sqrt{d_{\text{in}}}$, the scaling factor $1/s$ preserves activation variance across layers (LeCun et al., 1998; Glorot & Bengio, 2010; He et al., 2015; Karras et al., 2024), while expressing all weight tensors $\widetilde{W}$ at a common unit-variance scale independent of the input dimension. In addition, this parameterization equalizes effective learning rates.

Because activations and weights are expressed on comparable scales, a single bound c can be used for all. This gives c a simple interpretation as a shared dynamic-range constraint throughout the model, applying uniformly to residual activations and weight tensors alike (see Algorithm 1).

Under a standard normal initialization, choosing $c = 2.1$ places the clipping boundary slightly beyond two standard deviations and therefore leaves the vast majority of initial values unchanged. We further validate this choice by simulating Gaussian random walks with clipping (Fig. 6), where $c \in [1.8, 2.2]$ best preserves unit-variance over time. Alternative unit-variance initializations, such as Rademacher (Abbe et al., 2025) or Hadamard (Zhao et al., 2022), behave similarly to standard normal initialization.

For Adam-like optimizers (Kingma & Ba, 2015; Jordan et al., 2024), preserving the learning dynamics of the standard parameterization requires rescaling the optimizer hyperparameters as

$$\eta' := \eta \cdot s, \qquad \lambda' := \lambda / s, \tag{7}$$

where η and λ denote the learning rate and weight decay, respectively. Bias parameters, when present, retain their original hyperparameters. A derivation is given in Appendix E.

To enforce the exact bound $[-c, c]$, we clip the optimizer-facing parameters elementwise after each update (Courbariaux et al., 2015):

$$\widetilde{w}'_{ij} := \text{clip}(\widetilde{w}_{ij}, c). \tag{8}$$

This imposes a hard constraint on the parameter range and protects against occasional large updates.

3.3 Enhancements

The performance of our final model benefits from two enhancements. The complete noNorm model, including all introduced components, is given in pseudocode in Algorithm 1.

Depth-Dependent Input Scaling. By design, skip-link activations in our architecture grow at most linearly with depth, as enforced by Eq. (1). This differs from standard transformers, where such control generally holds only in expectation (Zhang et al., 2019; Hayou et al., 2021). At initialization, the variance of skip activations increases linearly with depth, since residual contributions accumulate along the skip path. Hence, we scale the residual-branch input $\mathbf{x}_\ell$ by $1/\sqrt{\ell + 1}$, computing $\mathbf{F}_\ell(\mathbf{x}_\ell/\sqrt{\ell + 1})$ instead of $\mathbf{F}_\ell(\mathbf{x}_\ell)$ in (1). Our ablation study shows that this improves the loss, although it is not required for stable training (see Appendix F).

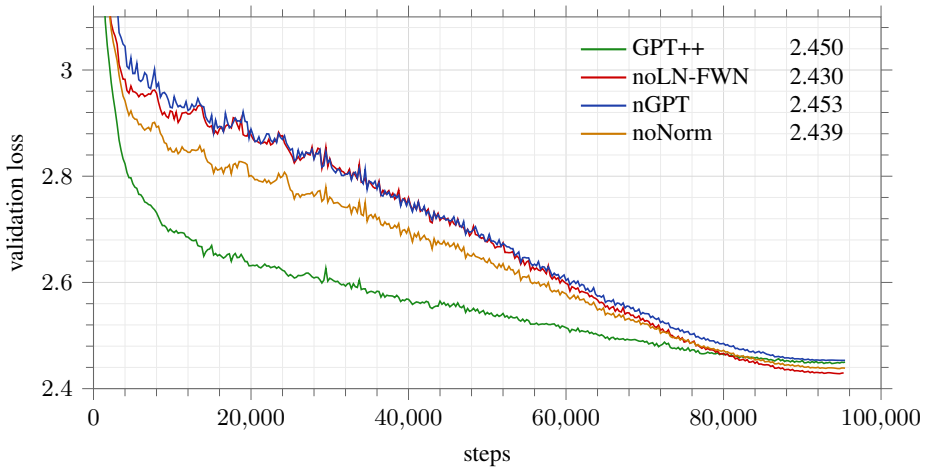


Figure 2: Validation loss on FineWeb-Edu for 600M-parameter models trained on 100B tokens.

Learnable Attention Temperature. Because the parameterization in (6) reduces the extent to which weight magnitudes can adapt during training, it also limits the effective softmax temperature. Since the sharpness of the softmax can affect attention quality, we introduce a learnable per-head temperature (Ryan, 2024). Concretely, for each head we use a parameter s'_h initialized to one, applying the same parametrization from (3), and scale the queries of head h as

$$q_h := s_h \cdot \mathrm{RoPE}(\mathbf{x}_h W_{Q,h}^\top) \tag{9}$$

4 Results

We apply our new normalization-free residual block (1) to language modeling and speech recognition, and study its activation and weight dynamics. Our results show that eliminating normalization matches or surpasses strong pre-norm baselines while producing better controlled internal dynamics that make the models exceptionally well suited for low-precision quantization.

4.1 Experiments

All language-model experiments use a nanoGPT-style codebase (Karpathy, 2022; 2025) and the FineWeb-Edu dataset (Penedo et al., 2024). All models use the GPT-2 vocabulary of size 50,304. Our primary baseline is GPT++, a pre-norm transformer with RMSNorm, RoPE with QK-Norm, a 2-layer MLP with ReLU^2 activation, logits soft-capping, and no biases (Touvron et al., 2023; Karpathy, 2025). The noNorm model does not have any normalization layers: RMSNorm layers at the start of residual blocks, after Q and K, and at the end of the network are removed. We also compare against nGPT (Loshchilov et al., 2025), an alternative approach of bounding weights and residuals that instead introduces additional normalizations of activations and weights. We use their publicly available model code and recommended hyperparameters. Finally, we benchmark against a model identical to noNorm but with forced weight normalization (FWN) (Karras et al., 2024) instead of weight clipping to enable direct comparison between clipping and normalization of weights. We refer to this last variant as 'noLN-FWN'.

Unless otherwise stated, models are trained with sequence length 1024 and $2^{20} = 1{,}048{,}576$ tokens per batch for a total of 100B tokens (95,430 training steps). For the 600M-parameter setting, this token budget is well above the Chinchilla-optimal ratio of approximately $20\times$ parameters. We use the Muon optimizer (Jordan et al., 2024) with Moonlight's learning-rate adjustment (Liu et al., 2025); as shown in the Appendix, the qualitative comparison is unchanged with AdamW (Appendix F). Details of the experimental configuration and hyperparameters can be found in Appendix D.

Autoregressive Language Models. Fig. 2 shows validation loss for 600M-parameter models trained on 100B tokens. noNorm, noLN-FWN, and nGPT initially underperform GPT++, but exhibit

Table 1: lm-eval results for 600M-parameter models trained on 100B tokens. **Bold**: best in column.

	Accuracy ↑							Perplexity ↓		
Model	ARC-C	ARC-E	HS	Lmb.	PIQA	WG	Avg	Lmb.	Wiki	Avg
GPT++	**36.86**	**66.20**	53.96	**45.24**	72.03	56.67	55.16	**14.67**	20.65	17.66
nGPT	34.90	63.72	52.74	43.47	71.00	55.56	53.57	16.43	20.64	18.54
noLN-FWN	36.69	65.95	**54.50**	44.79	**72.47**	56.20	55.10	14.86	**19.99**	**17.42**
noNorm	36.60	65.99	54.25	45.16	71.60	**58.17**	**55.30**	14.79	20.23	17.51

steeper loss trajectories during training and ultimately achieve comparable or lower final loss values. This pattern aligns with the results reported by Fu et al. (2025). Compared to the results of the original study, nGPT in our experiments shows less benefit over GPT++. This may be due to the different optimizer. Furthermore, when comparing variants with weight clipping and weight normalization, the latter achieves a slightly lower loss.

Table 1 reports downstream evaluation on lm-eval-harness tasks (Gao et al., 2024). We report answer-length-normalized accuracy for ARC-Challenge (ARC-C), ARC-Easy (ARC-E) (Clark et al., 2018), HellaSwag (HS) (Zellers et al., 2019), PIQA (Bisk et al., 2020), Lambada (Lmb.) (Paperno et al., 2016), and WinoGrad (WG) (Levesque et al., 2012). We also report perplexity on Lambada (Lmb.) (Paperno et al., 2016) and Wikitext (Wiki) (Merity et al., 2017). Overall, models perform similarly on downstream tasks despite some differences in loss. noNorm is competitive with GPT++ and nGPT, and also matches noLN-FWN, which uses weight normalization instead of weight clipping.

Table 2: Block-diffusion language modeling. noNorm outperforms the GPT++ baseline across all model sizes in FineWeb-Edu perplexity. Models are trained on 10B tokens; **Bold**: best in column.

Model	Perplexity (75M) ↓	Perplexity (150M) ↓	Perplexity (600M) ↓
GPT++	53.54	25.75	20.88
noNorm	**50.32**	**24.13**	**19.64**

Diffusion Language Models. We also test noNorm on block diffusion (Arriola et al., 2025), a recent diffusion language model that is competitive with autoregressive baselines. Although block diffusion uses a different training objective and masking structure, noNorm outperforms the pre-norm GPT++ baseline across all tested model sizes. The results in Table 2 suggest that the advantages of norm-free scale control are not limited to standard autoregressive training.

Speech Recognition. To evaluate if normalization-free design principles can be easily applied to a very different task, we train Fast-ConformerL (Rekesh et al., 2023) on LibriSpeech (Panayotov et al., 2015), without any architecture-specific considerations. We train for 100 epochs using NeMo (Kuchaiev et al., 2019). The Fast-Conformer architecture combines attention, MLP, and convolutional residual blocks with downsampling and sparse attention. We simply remove all normalization layers on the skip path and at the start of residual blocks, implement Eq. (1) and apply Eq. (5) for activation clipping at residual block outputs and for weight clipping. We scale learning rate and weight decay as in (7). The noNorm Fast-ConformerL attains a word-error rate (WER) of 8.35%, quite close to the baseline WER of 8.21%. This can likely be improved further.

4.2 Activation and Weight Dynamics

Across activations and weights, the pre-norm model consistently develops much larger dynamic ranges, whereas the noNorm model remains controlled across network depth and throughout training.

Activations. Fig. 3 analyzes trained 600M parameter models. In the pre-LN model, the RMS of skip-path activations grows from about 0.03 at the embedding to 12.7 in the final block, an approximately $400\times$ increase. By contrast, in noNorm the skip-path RMS grows far more gradually, approximately linearly with depth, which is consistent with the bounded residual-branch outputs.

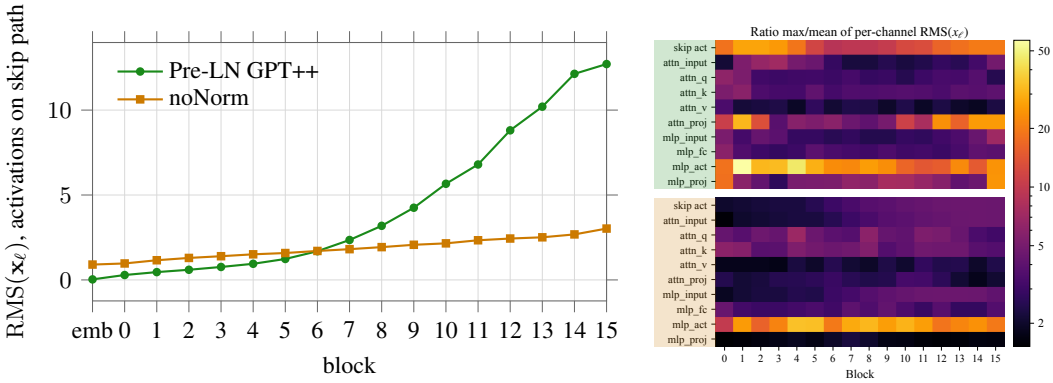


Figure 3: Activation analysis for 600M-parameter models trained on 100B tokens. **Left:** Skip-path activation RMS across network depth. **Right:** outlier-channel severity across the network, measured as $\max/\text{mean}$ of per-channel RMS. Pre-LN (green, top) develops substantially larger activation growth and more severe outliers than noNorm (orange, bottom).

This difference is accompanied by markedly stronger activation outliers in the pre-LN model, both outlier channels and individual outlier activations. Outlier channels are most prominent on the skip path and at the outputs of layers mapping back to the skip path space (`attn_proj` and `mlp_proj`), whereas noNorm shows both fewer outlier channels and less severe ones. A more detailed analysis of activations during training is provided in Section H.1, while Section H.3.1 analyzes outliers.

Weights. A similar improvement can be observed for the evolution of the weights during training as illustrated in Fig. 24. In noNorm, weight variances remain close to their initialized scale throughout training, whereas the pre-LN model shows much larger variation over time and across layers. Likewise, the relative weight update magnitude, $\|\Delta w\|/\|w\|$, closely follows the cosine-shaped learning-rate schedule in noNorm but not in GPT++.

This behavior supports the interpretation of Section 3.2: by keeping both activation and weight scales controlled, noNorm makes the effective learning rate more uniform and more directly governed by the optimizer schedule (see Figure 14 and Appendix H.2). This avoids obscured and unpredictable learning dynamics. It also avoids unexpectedly large parameter updates, so tricks like Adaptive Gradient Clipping are not required (Brock et al., 2021). A detailed analysis per layer, including evolution of weights, weight updates, learnable scales during training is deferred to Section H.2.

Table 3: Validation loss for 600M-parameter models with weight + activation post-training quantization (weights per-tensor symmetric, activations per-token, round-to-nearest, no calibration or finetuning). Δ is the degradation over BF16.

		INT8		INT4	
Model	**BF16**	**W8 Δ**	**W8A8 Δ**	**W4 Δ**	**W4A4 Δ**
GPT++	2.450	+0.019	+0.021	+10.121	+9.938
nGPT	2.453	+0.001	+0.003	+0.490	+1.311
noLN-FWN	2.430	+0.003	+0.004	+1.396	+5.665
noNorm	2.439	+0.000	+0.001	+0.031	+0.510

4.3 Robust Quantization

We evaluate post-training quantization robustness on the 600M-parameter models using emulated quantization, without any finetuning. We test weight-only quantization (W8 and W4), as well as joint weight-and-activation formats (W8A8 and W4A4). As shown in Table 3, noNorm exhibits the strongest quantization robustness among the compared models and can be quantized to 4-bit weights with only a marginal increase in loss, whereas GPT++ fails entirely. noLN-FWN and nGPT perform much better, with nGPT being more robust to quantization than noLN-FWN, but

much less than noNorm. The robustness of noNorm follows directly from elementwise bounding of activations and weights, which minimizes the dynamic-range mismatches and outliers that typically drive quantization error. This is *not* the case with vector normalization, which provides a norm but not an element-wise bound.

Weight regularization can shape the weight distribution, which in turn determines how amenable the weights are to quantization. For example, with low weight decay weights accumulate near the boundaries of the allowed interval. With larger weight decay, the distribution becomes more Gaussian (see Fig. 7), and is thus harder to quantize: a fivefold increase in weight decay results in a $10\times$ larger quantization error even though the BF16 loss is similar. We discuss this relationship further in Appendix E.2.

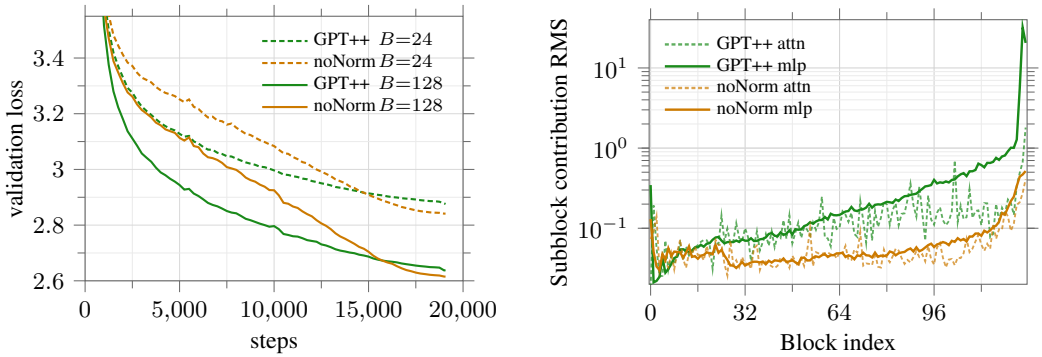


Figure 4: Scaling up the 150M parameter base models by increasing the number of blocks from 24 to 128. **Left**: validation loss. **Right**: RMS of attn and mlp residuals for B=128

Table 4: Summary of the most informative 150M/10B ablations. Deltas are measured relative to the default noNorm configuration (validation loss 2.9111). Complete results are reported in Section F.

Category	Setting	Val. loss ↓	Δ
Depth-dependent input scaling	enabled	2.8700	-0.0411
Weight decay	$\lambda = 0.01$	2.8856	-0.0255
Activation clamping	disabled	diverged	N/A
Weight clamping function	tanh	2.9613	+0.0502
Scale parametrization	linear	2.9332	+0.0221

4.4 Ablation Studies

To identify the most important design choices, we run controlled ablations from a minimal noNorm configuration (Table 6). These experiments use 150M-parameter models trained on 10B tokens.We report the most important findings here and defer the complete set of ablations to Appendix F.

Model design. Training diverges when all activation clipping is removed. Clipping only at the residual-branch output gives the lowest loss among the tested locations. Weight clipping with a well-chosen threshold further improves full-precision loss and substantially reduces degradation after 4-bit quantization. Weights and activations should to be bounded (see clipping details below). Analyses confirm more uniform weight updates and fewer, less severe activation outliers for noNorm. The output scale α should be initialized sufficiently small. A significant improvement comes from adding depth-dependent input scaling, with smaller gains from adding query-head scales. For small 150M models, high weight decay improves loss. Overall, training of the noNorm model is robust and benefits from the explicit control of residual scale and optimization dynamics.

Scaling At 150M parameters and 10B tokens, with a depth of 24 blocks, noNorm performance approximately matches GPT++. At 600M parameters, noNorm achieves better loss. Moreover, noNorm can be scaled from 24 to 128 blocks with no hyperparameter changes (Figure 4 and Table 15).

Clipping. Figure 5 traces the origins of the quantization benefits to the architectural modifications of noNorm: varying the clipping thresholds for both weights and activations reveals their respective

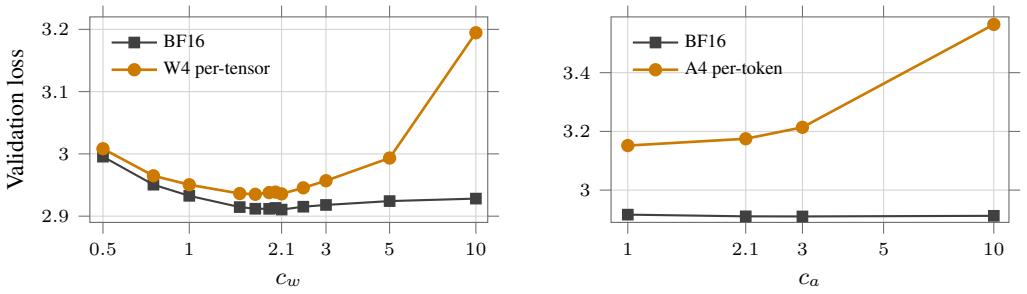


Figure 5: Effect of weight (left) and activation (right) clipping threshold on loss when quantizing.

impact on quantizability. For weights, there is an optimal weight clipping value of 2.1. Below, weights are too constrained and above, outliers drastically harm the quantization performance. As expected, looser weight bounds generally lead to less controlled effective learning rates and also increase the BF16 loss (see Figure 14 and Appendix H.2). For activations, when quantizing to 4 bits tighter residual output clipping is also better, with best BF16 loss around 2.1. Comparing clipping methods, weight quantization benefits from KKT clipping, as Table 10 shows. Interestingly, KKT-based weight clipping also improves activation quantization. For activations, naive hard clipping performs similarly to KKT clipping, but it does result in larger activation RMS (Figure 10).

5 Limitations and Future Work

Our focus on normalization and boundedness leaves some directions open for further work. noNorm matches GPT++ but is not much better in terms of loss at full precision. Its advantage lies in the fact that it is much easier to reason about and that it is much easier to quantize. Isolating the core mechanisms, we did not evaluate very large models, and our experiments are confined to transformer-style architectures for language and speech tasks. As the reparameterization alters the scale of the optimized weights, hyperparameters may need to be rescaled, too. Some existing models have many separately tuned learning rates or weight decay for different parameters, which can be tricky to adapt to our reparametrization.

Weight normalization can result in better full precision loss than weight clipping but is harder to quantize. It may be that our clipping algorithm can be improved, or that other properties of normalization are responsible for the difference. One potential future avenue is to combine normalization operations with elementwise bounding for better quantization. Another direction for future work is to explore specialized techniques for shaping the weight distribution to make it even more amenable to very low-precision quantization.

Our ablation studies show that replacing per-block scales α_ℓ with a single global scale α degrades loss only marginally. This simplification connects naturally to the continuous-time view of residual networks (Haber & Ruthotto, 2017), where a global α acts as a uniform integration step size Δt in a Neural ODE. In this view, our initialization ensures a stable discretization, while clipping bounds the vector field $\mathbf{F}$ to prevent numerical explosion during Euler integration. Exploring this theoretical link is a promising direction for future work.

6 Conclusion

We show that stable training of deep residual networks does not inherently depend on normalization but instead on explicit control of dynamic range. By framing activation and weight bounding as a constrained nonlinear optimization problem, we derive a simple, normalization-free formulation based on element-wise clipping that provides control over weight and activation dynamics during training. The resulting bounded computations make this approach especially attractive for quantization and low-precision deployment, and suggest a practical path toward simpler and more efficient large language models.

7 AI USE STATEMENT

In this work, generative AI tools were used only to assist with writing, formatting, editing, and minor coding tasks. No part of the core methodology or scientific contribution was autonomously generated by an LLM.

We used generative AI tools for providing feedback on research methodology and experiments, to support data analysis via visualizations, to help survey related work, and to improve readability of the manuscript. We did not use generative AI tools for generating data, to write the core training or model code, to write proofs or mathematical derivations, or for managing datasets.

Specifically, we used generative AI tools for writing Python scripts that collect log data from multiple runs into DataFrames that can be used for creating LaTeX tables or visualizations, brainstorming and writing scripts and notebooks to quickly evaluate ideas, and to write scripts to interact with cluster environments. AI was used to write Python scripts that generate figures, tables or other artifacts. In this manner, we could review the code to verify correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

The paper includes detailed pseudo code for the proposed model architecture (Algorithm 1) and a complete description of all hyperparameters (Section D). We use the publicly available FineWeb-Edu dataset and nanoGPT as the training framework, with default data splits. While we do not provide access to executable code, the pseudo code is straightforward to implement in any standard framework. We additionally run extensive ablations over hyperparameters and model configurations to support the main claims. We report the GPU type, count, memory, and training time in Section D. The total compute across all reported experiments is estimated at 6400 GPU-hours.

9 ETHICS STATEMENT

This paper proposes a model architecture and provides a focused empirical analysis. We use only standard, publicly available academic datasets and do not involve human subjects or participants. While we do not foresee new risks introduced by this method, foundational research can have a wide range of uses in subsequent works that map it to practical applications. The models presented are far from deployment-ready since they are very small by today's standards, trained only on a small dataset, and have no post-training applied which is required for real-world use. The methods described could potentially be used to improve model quantization, thereby lowering energy consumption. We are not aware of any ethical concerns arising from this work with respect to privacy, bias, safety, or misuse.

REFERENCES

Emmanuel Abbe, Elisabetta Cornacchia, Jan Hązła, and Donald Kougang Yombi. Learning high-degree parities: The crucial role of the initialization. In *International Conference on Learning Representations*, volume 2025, pp. 19448–19494, 2025.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 06–11 Aug 2017.

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

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

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. ReZero is all you need: fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239. Dataset license: AFL-3.0. Link to data: https://yonatanbisk.com/piqa/.

Charles Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Prince, Björn Deiseroth, Andres Felipe Cruz Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μP: The unit-scaled maximal update parametrization. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 91057–91105, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e3130a164f5c724e37271b93bc76dd28-Paper-Conference.pdf.

Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1059–1071. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/brock21a.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, 2018. URL https://arxiv.org/abs/1803.05457. Dataset license: CC BY-SA 4.0. Link to data: https://huggingface.co/datasets/allenai/ai2_arc.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3123–3131, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/3e15cc11f979ed25912dff5b0669f2cd-Abstract.html.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 2793–2803, 2021.

Yonggan Fu, Xin Dong, Shizhe Diao, Matthijs Van keirsbilck, Hanrong Ye, Wonmin Byeon, Yashaswi Karnati, Lucas Liebenwein, Maksim Khadkevich, Alexander Keller, Jan Kautz, Yingyan (Celine) Lin, and Pavlo Molchanov. Nemotron-Flash: towards latency-optimal hybrid small language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 161342–161366. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ec2691f6b5c88eb650554dffbc0e0af5-Paper-Conference.pdf.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://github.com/EleutherAI/lm-evaluation-harness.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 97114–97144, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f1b04face60081b689ba740d39ea8f37-Paper-Conference.pdf.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34 (1):014004, dec 2017. doi: 10.1088/1361-6420/aa9a90. URL https://doi.org/10.1088/1361-6420/aa9a90.

Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable ResNet. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1324–1332. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/hayou21a.html.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L. Smith, and Yee Whye Teh. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 4475–4483, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024. Online technical report.

Andrej Karpathy. nanoGPT. https://github.com/karpathy/nanoGPT, 2022.

Andrej Karpathy. nanochat: The best ChatGPT that $100 can buy. https://github.com/karpathy/nanochat, 2025.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24174–24184, 2024.

William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, University of Chicago, Dept. of Mathematics, 1939. URL https://catalog.lib.uchicago.edu/vufind/Record/4111654.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, Patrice Castonguay, Mariya Popova, Jocelyn Huang, and Jonathan M. Cohen. NeMo: a toolkit for building AI applications using neural modules, 2019. URL https://arxiv.org/abs/1909.09577.

Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492. University of California Press, 1951.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Genevieve B. Orr and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pp. 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi: 10.1007/3-540-49430-8_2. URL https://doi.org/10.1007/3-540-49430-8_2.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2012. URL https://cdn.aaai.org/ocs/4492/4492-21843-1-PB.pdf. Dataset license: CC BY 4.0. Link to data: https://cs.nyu.edu/~davise/papers/WinogradSchemas/WSCollection.html.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe. Dataset license: CC BY-SA 3.0. Link to data: https://huggingface.co/datasets/Salesforce/wikitext.

Paul Merolla, Rathinakumar Appuswamy, John V. Arthur, Steve K. Esser, and Dharmendra Modha. Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv preprint arXiv:1606.01981*, 2016. doi: 10.48550/arXiv.1606.01981.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, NY, 2nd edition, 2006.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27198–27211. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ae0cba715b60c4052359b3d52a2cff7f-Paper-Conference.pdf.

NVIDIA, :, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. NVIDIA Nemotron 3: Efficient and Open Intelligence, 2025. URL https://arxiv.org/abs/2512.20856.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964. Dataset license: CC BY 4.0. Link to data: https://www.openslr.org/12/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/. Dataset license: CC BY 4.0. Link to data: https://zenodo.org/records/2630551.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf. Dataset license: ODC-BY 1.0. Link to data: https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Yuval Ran-Milo. Attention sinks are provably necessary in softmax transformers: Evidence from trigger-conditional tasks. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 70–89, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-391-3. doi: 10.18653/v1/2026.acl-short.8. URL https://aclanthology.org/2026.acl-short.8/.

Dima Rekesh, Nithin Rao Koluguri, Samuel Kriman, Somshubra Majumdar, Vahid Noroozi, He Huang, Oleksii Hrinchuk, Krishna Puvvada, Ankur Kumar, Jagadeesh Balam, and Boris Ginsburg. Fast conformer with linearly scalable attention for efficient speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, 2023. doi: 10.1109/ASRU57964.2023.10389701.

Nick Ryan. Introducing a learnable temperature value into the softmax self-attention scores. https://nickcdryan.com/2024/08/02/introducing-a-learnable-temperature-value-into-the-self-attention-scores/, 2024. Accessed: 2026-04-30.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/ed265bc903a5a097f61d3ec064d96d2e-Paper.pdf.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. doi: 10.48550/arXiv.2002.05202. URL https://arxiv.org/abs/2002.05202.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=F7aAhfitX6.

Justin Teo and Jonathan P. How. Anti-windup compensation for nonlinear systems via gradient projection: Application to adaptive control. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 6910–6916, 2009. doi: 10.1109/CDC.2009.5400075.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42, 2021.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Mitchell Wortsman, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Replacing softmax with ReLU in vision transformers. *arXiv preprint arXiv:2309.08586*, 2023. doi: 10.48550/arXiv.2309.08586. URL https://arxiv.org/abs/2309.08586.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/. Dataset license: MIT. Link to data: https://huggingface.co/datasets/Rowan/hellaswag.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M. Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 40770–40803, 2023.

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 (eds.), *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.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gsz30cKX.

Jiawei Zhao, Florian Tobias Schaefer, and Anima Anandkumar. ZerO initialization: Initializing neural networks with only zeros and ones. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=1AxQpKmiTc.

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14901–14911, June 2025.

A PSEUDOCODE OF NONORM TRANSFORMER

Algorithm 1: Pseudocode of our noNorm model. We omit RoPE and attention heads for brevity.

Input: token sequence $\mathbf{x} \in \mathbb{N}_0^T$
Output: logits $y \in \mathbb{R}^{T \times V}$
Parameters: $W_{\text{embed}}, W_{\text{unembed}}$, initialize to $\mathcal{N}(0, 1)$
Hyperparameters: $c := 2.1$

Module `ClipProj`
  function `forward`($\mathbf{x}$)
    return $\text{clip}(\mathbf{x}, -c, c)$
  function `backward`($\nabla\mathbf{y}, \mathbf{x}$)
    return $\nabla\mathbf{y} \cdot \mathbf{1}[|\mathbf{x}| < c \ \vee \ \nabla\mathbf{y} \cdot \mathbf{x} \geq 0]$

function `explinear`(x)
  if $x < 1$ **then return** e^{x-1} **else return** x

Module `BoundedLinear`
  function `forward`($\mathbf{x}, W$)
    return $\mathbf{x} \cdot \left(\texttt{ClipProj}(W)/\sqrt{d_{\text{in}}}\right)^\top$

Module `Attention`
  Parameters: $W_q, W_k, W_v, W_{\text{proj}}$, initialize to $\mathcal{N}(0, 1)$,
    s'_q, initialize to 1 (per head)
  function `forward`($\mathbf{x}$)
    $\mathbf{q} \leftarrow \texttt{BoundedLinear}(\mathbf{x}, W_q)$
    $\mathbf{k} \leftarrow \texttt{BoundedLinear}(\mathbf{x}, W_k)$
    $\mathbf{v} \leftarrow \texttt{BoundedLinear}(\mathbf{x}, W_v)$
    $\mathbf{q} \leftarrow \texttt{RoPE}(\mathbf{q}) \cdot \texttt{explinear}(s'_q)$
    $\mathbf{k} \leftarrow \texttt{RoPE}(\mathbf{k})$
    $\mathbf{x} \leftarrow \texttt{CausalSDPA}(\mathbf{q}, \mathbf{k}, \mathbf{v})$
    return `BoundedLinear(`$\mathbf{x}$, W_{proj}`)`

Module `MLP`
  Parameters: $W_{\text{fc}}, W_{\text{proj}}$, initialize to $\mathcal{N}(0, 1)$
  function `forward`($\mathbf{x}$)
    $\mathbf{x} \leftarrow \texttt{BoundedLinear}(\mathbf{x}, W_{\text{fc}})$
    $\mathbf{x} \leftarrow \text{ReLU}^2(\mathbf{x})$
    return $\texttt{BoundedLinear}(\mathbf{x}, \ W_{\text{proj}})$

Module `Block`
  Parameters: $\alpha'_{\text{attn}}, \alpha'_{\text{mlp}}$, initialize to -3.6, so that $\alpha_{\text{attn}}, \alpha_{\text{mlp}}$ are initially 0.01
  function `forward`($\mathbf{x}$, l)
    $\alpha_{\text{attn}} = \texttt{explinear}(\alpha'_{\text{attn}})$
    $\alpha_{\text{mlp}} = \texttt{explinear}(\alpha'_{\text{mlp}})$
    $\mathbf{x} \leftarrow \mathbf{x} + \texttt{ClipProj}(\alpha_{\text{attn}} \cdot \texttt{Attention}(\mathbf{x}/\sqrt{2l+1}))$
    $\mathbf{x} \leftarrow \mathbf{x} + \texttt{ClipProj}(\alpha_{\text{mlp}} \cdot \texttt{MLP}(\mathbf{x}/\sqrt{2l+2}))$
    return $\mathbf{x}$

$\mathbf{x} \leftarrow \texttt{Embed}(\mathbf{x}, W_{\text{embed}})$
$\mathbf{x} \leftarrow \texttt{ClipProj}(\mathbf{x})$
for $l = 0$ **to** $L - 1$ **do**
  $\mathbf{x} \leftarrow \texttt{Block}_l(\mathbf{x}, l)$
$\mathbf{y} \leftarrow \texttt{BoundedLinear}(\mathbf{x}, W_{\text{unembed}})$
return $15 \cdot \tanh(\mathbf{y}/15)$

B BASELINE TRANSFORMER

The standard GPT++ pre-norm model is a decoder-only transformer (Touvron et al., 2023) which first maps token indices $(t_1, \ldots, t_T)$ to vectors $x^{(0)} \in \mathbb{R}^{T \times d}$ via an embedding table $E \in \mathbb{R}^{V \times d}$, initialized as $E_{ij} \sim \mathcal{N}(0, 0.01^2)$. The token vectors are then passed through L transformer blocks, each applying attention and MLP with pre-norm residual connections, and finally passed through an unembedding $U \in \mathbb{R}^{d \times V}$ to obtain logits, which are bounded by elementwise soft-capping $\mathrm{softcap}_{15}(z) := 15 \tanh(z/15)$ before being converted to probabilities by softmax. The model does not use biases.

$$
\begin{aligned}
x^{(0)} &= E(\text{token}_{\text{indices}}) \\
&\text{for } \ell = 1 \ldots L: \\
\hat{x}^{\ell} &= x^{(\ell-1)} + \text{Attn}_l\left(\gamma_{l,attn}\,\mathrm{RMSNorm}(x^{(\ell-1)})\right) \\
x^{(\ell)} &= \hat{x}^{\ell} + \text{MLP}_l\left(\gamma_{l,mlp}\,\mathrm{RMSNorm}(\hat{x}^{\ell})\right) \\
y &= \left(\gamma_{pred} \odot \mathrm{RMSNorm}(x^{(L)})\right)U \\
p_{\text{pred}} &= \mathrm{softmax}(\mathrm{softcap}_{15}(y))
\end{aligned}
$$

where $\mathrm{RMSNorm}(x) = x/\sqrt{\frac{1}{d}\sum_i x_i^2 + \varepsilon}$ (with $\varepsilon = 10^{-6}$) and the learnable gains $\gamma \in \mathbb{R}^d$ are initialized to 1. We explicitly separate the learnable scales from the normalization operation for clarity, as opposed to the original definition of RMSNorm(Zhang & Sennrich, 2019).

The Attention block consists of multi-head causal scaled dot-product self-attention (sdpa) with H heads and per-head dimension $d_h = d/H$, (Su et al., 2024) and QK-norm (Henry et al., 2020). In the following attention equations, z is the input of the residual block with the normalization layer applied.

$$
\begin{aligned}
q, k, v &= zW_Q^\top,\ zW_K^\top,\ zW_V^\top \\
\hat{q}, \hat{k} &= \text{RMSNorm}(\text{RoPE}(q)),\ \text{RMSNorm}(\text{RoPE}(k)) \\
\text{sdpa}(z) &= \text{softmax}(\hat{q}\hat{k}^\top/\sqrt{d_h})\,v \\
\text{Attn}(z) &= \text{sdpa}(z) \cdot W_O^\top
\end{aligned}
$$

The MLP block can be a 2-layer MLP, where the function 'act' may be any of e.g. SiLU, GELU, ReLU or ReLU^2.

$$\text{MLP}(z) = \text{act}(zW_{\text{up}}^\top)W_{\text{down}}^\top$$

or a Gated Linear Unit (GLU) (Shazeer, 2020), where the hidden dimension is 2/3 of the 2-layer MLP hidden-dimension to match parameter counts:

$$\text{MLP}(z) = (\text{SiLU}(zW_{\text{gate}}^\top) \odot zW_{\text{up}}^\top)\,W_{\text{down}}^\top$$

Recent studies find ReLU^2 performs similar to GLU-style MLPs while being more hardware-efficient so we adopt that for our GPT++ baseline (Karpathy, 2025; NVIDIA et al., 2025).

Embeddings are initialized as $\mathcal{N}(0, 0.01^2)$, Linear layers as $W_{ij} \sim \mathcal{N}(0, 0.02^2)$. Output projections ($W_O$, W_{down}) use $\sigma = 0.02/\sqrt{2L}$ so each block contributes $O(1/\sqrt{L})$ to the residual (Radford et al., 2019)

C HIGH-LEVEL COMPARISON OF GPT++ AND NONORM

	GPT++ Section B	noNorm Algorithm 1
Initialization	$N(0, 0.01^2)$ for embeddings, $N(0, 0.02^2)$ for weights, $\sqrt{2L}$ smaller for output projections	$N(0,1)$, $\alpha_l = 0.01$ (3), $\mathbf{s}_h = 1$ (9)
Branch input	RMSNorm with learned scales	divide by $\sqrt{2l+1}$ before attention, divide by $\sqrt{2l+2}$ before the MLP
attention Q/K	Per-head RMSNorm	Learned per-head query scale
Linear layer weights	Directly trained	Optimizer sees $\widetilde{W}$; effective $W = clip(\widetilde{W}, c)/\sqrt{d_{\text{in}}}$
Residuals	Added directly	Scaled by learned α_ℓ, then clipped elementwise(5), then added (1)
Network output	RMSNorm with learned scales before unembedding, logits softcap	Logits softcap
Training	Unbounded weights and embeddings	Weights and embeddings constrained to $[-c, c]$ Sections 3.1 and 3.2

Table 5: Overview of differences between GPT++ and noNorm transformers. Both use causal softmax attention, RoPE, and a ReLU^2 MLP. Here $l \in [0, L-1]$ with L the number of blocks, and $c = 2.1$.

D HYPERPARAMETERS AND BASELINES

All models use the Muon optimizer (Jordan et al., 2024) with Moonlight's LR adjustment (Liu et al., 2025), gradient clipping at 1.0, sequence length 1024, ReLU^2 MLP activations (no GLU), RoPE (Su et al., 2024), and a logit soft cap $15 \cdot \tanh(\cdot/15)$. The LR schedule is linear warmup (715 steps) followed by cosine decay to 10^{-4}. We show that run-to-run variance is very low in Section F, and therefore the main 600M comparisons and most ablations use a single training run each. Full hyperparameters for the three main configurations are in Table 6. We provide details for the other 600M-scale baselines below. Note the 150M noNorm configuration is only the ablation baseline and not optimal (Section F); the 600M configuration is recommended.

noNorm. Our model is described in the main text and in Algorithm 1. We use the same hyperparameters as GPT++, scaled according to Eq. (7). We did not do any learning rate tuning, but did evaluate higher weight decay of 0.005 and 0.01 for the 600M model which were worse than the 0.002 obtained by Eq. (7).

noLN-FWN. noLN-FWN is the same as noNorm except that we do not use weight clipping but apply Forced Weight Normalization (FWN) (Karras et al., 2024) to all weight matrices and embeddings: every weight matrix row is renormalized after each optimizer step to uncentered variance 1. Hyperparameters are the same as noNorm, except weight decay is 0.

nGPT. nGPT (Loshchilov et al., 2025) also starts from the same base architecture but constrains all weights and token vectors to the unit hypersphere by L2-normalizing intermediate activations in many locations, and additionally L2-normalizing weight matrices after every optimizer step. The difference to FWN is that the normalization is applied only once after the optimizer step, not during the forward pass, and that weights are normalized to L2-norm of 1.0 instead of uncentered variance of 1.0, a difference of $\sqrt{d_{model}}$. nGPT also replaces the standard residual connection with a normalized interpolation: $h \leftarrow \text{normalize}\big(h + \alpha\big(\text{normalize}(\Delta h) - h\big)\big)$ where per-block learnable step sizes α_{attn}, α_{mlp} control update magnitude. We use the original publicly available implementation with recommended hyperparameters: peak learning rate 10^{-3}, weight decay 0, and no warmup. We swept learning rates $\{5\times10^{-4}, 10^{-3}, 2\times10^{-3}, 3\times10^{-3}\}$ and found 10^{-3} results in the lowest loss.

Compute. Ablation runs used 8×L40S GPUs (48GB VRAM each, approx. 4h per run for 10B tokens). Large 600M runs used 16×L40S (approx. 60h). The total compute for all reported experiments is estimated at 6400 GPU-hours.

Table 6: Hyperparameters for the three main configurations. Shared across all runs: micro-batch 16, Muon, gradient clip 1.0, seq. len. 1024, linear-warmup + cosine-decay LR, ReLU^2 MLP (no GLU), logit soft cap $15\cdot\tanh(\cdot/15)$. "—" indicates not applicable.

Setting	GPT++ 600M	noNorm 600M	noNorm 150M (ablation baseline)
Training			
Dataset	`edu_fineweb100B`	`edu_fineweb100B`	`edu_fineweb10B`
Steps	95 430	95 430	19 073
Tokens / batch	1 048 576	1 048 576	524 288
Peak LR	10^{-3}	5×10^{-2}	5×10^{-2}
Min LR	10^{-4}	10^{-4}	10^{-4}
Weight decay	0.1	0.002	0.002
Architecture			
Layers	16	16	24
Width	1536	1536	512
Heads	16	16	8
MLP dim	6144	6144	4096
Tied embeddings	No	No	Yes
Normalization / bounding			
Pre-norm / final norm	RMS / RMS	— / —	— / —
Depth-dep. residual rescale ($\div\sqrt{\ell+1}$)	—	Yes	No
Learned output scale (init)	—	Yes (0.01)	Yes (0.01)
Embedding FWN	No	No	Yes
Activation clamp (loc. / fn / threshold)	—	output / onesided / 2.1	output / onesided / 2.1
Weight clamp (fn / threshold)	—	onesided / 2.1	onesided / 2.1

E Hyperparameter Scaling for Weight-Rescaled Layers

The following analysis is very similar to the derivation in (Karras et al., 2024) and (Loshchilov et al., 2025), but we show it here for noNorm for completeness. noNorm stores every matmul weight at unit variance and applies a $1/s$ forward-pass multiplier with $s_\ell = \sqrt{\mathrm{d_in}_\ell}$ per layer, so $W_{\text{eff}} = W'/s$ and $y = (1/s)\,W'x + b$. Forward outputs are identical to the standard parameterization (SP) with $W \sim \mathcal{N}(0, 1/\mathrm{fan_in})$, but AdamW now updates W' instead of W_{eff}. This presents us with the opportunity to equalize effective learning rates across layers without requiring layer-dependent hyperparameter scaling: because every W' has unit scale, the AdamW update $\|\Delta W'\|/\|W'\| = \mathcal{O}(\eta)$ (Kingma & Ba, 2015) is independent of $\mathrm{fan_in}$ — wider and narrower layers get the same relative update magnitudes. This means that hyperparameters must change if we want to preserve the same training behaviour as in SP.

Since $W_{\text{eff}} = W'/s$, the chain rule gives $\nabla_{W'}\mathcal{L} = (1/s)\,\nabla_{W_{\text{eff}}}\mathcal{L}$, so Adam's moments propagate as $\hat{m}' = \hat{m}/s$ and $\hat{v}' = \hat{v}/s^2$. The Adam direction on W' is then

$$\frac{\hat{m}'}{\sqrt{\hat{v}'}+\varepsilon'} = \frac{\hat{m}/s}{\sqrt{\hat{v}}/s+\varepsilon'} = \frac{\hat{m}}{\sqrt{\hat{v}}+s\varepsilon'},$$

which reproduces the SP direction $\hat{m}/(\sqrt{\hat{v}}+\varepsilon)$ when $\varepsilon' = \varepsilon/s$. Matching the effective-weight update $\Delta W_{\text{eff}} = \Delta W'/s$ term by term against SP's update — both the Adam step (fixing $\eta'/s = \eta \Rightarrow \eta' = s\eta$) and the decoupled weight decay (fixing $\eta'\lambda' = \eta\lambda \Rightarrow \lambda' = \lambda/s$) — gives

$$\eta' = s\,\eta, \qquad \lambda' = \lambda/s, \qquad \varepsilon' = \varepsilon/s \qquad \text{(AdamW)}. \tag{10}$$

Under SGD there is no $\sqrt{\hat{v}}$ normalization, so a second power appears in η':

$$\eta' = s^2\,\eta, \qquad \lambda' = \lambda/s^2 \qquad \text{(SGD)}. \tag{11}$$

The hyperparameters for biases are not rescaled, since the reparameterization does not affect these. We numerically verified that the scaling Eq. (10) yields the exact same training trajectory as SP over 200 training steps; omitting the ε correction gives $\sim 10^{-6}$ relative error per step, omitting the λ correction gives $\sim 5\times10^{-3}$.

Matching behaviour of SP is straightforward if the original model uses the same initialization for all its layers, as for our baseline pre-LN GPT++, which uses $N(0, 0.02^2)$. However, if the original model used dimension-dependent initialization, as e.g. $N(0, 1/\mathrm{fan}_{in})$, matching SP behaviour would require scaling hyperparameters individually depending on every matrix shape.

The noNorm model has different training behaviour than pre-LN GPT++ by design (due to removing normalization, bounding activations, bounding weights), so it is difficult to exactly match standard GPT++ training behaviour. However, the learning rate analysis still applies and is useful for hyperparameter transfer from SP to unit-scale reparameterized models. Our baseline pre-LN GPT++ uses $N(0, 0.02^2)$ for initialization instead of $1/\sqrt{D}$, so we use that factor of 50 compared to noNorm which uses $N(0, 1)$. This makes it easy to match learning rates, as no per-layer scaling is required: we simply scale learning rate with that factor 50 from 0.001 to 0.05, and decrease weight decay with the same factor from 0.1 to 0.002.

Interaction with weight bounding. Because Adam's relative per-step update is $\mathcal{O}(\eta/\|W'\|)$, an increase in $\|W'\|$ results in smaller relative weight updates $\|\Delta w\|_F/\|w\|_F$, i.e. different effective learning rates across layers, especially as some weights grow during training (Section H.2). NoNorm's element-wise clipping of w_{ij} prevents weights from growing too much, placing a lower bound on the effective learning rate so that this effect is significantly reduced. This also means that effective learning rates closely follow the learning rate schedule as demonstrated in Section H.2. This now allows for easy control over how much a parameter changes per training step. This would still apply without the weight-reparameterization: the equivalent clip threshold without reparameterization would be a per-layer clipping threshold $c/\sqrt{d_{in}}$.

Empirical validation of effective learning rates Though the original AdamW paper (Loshchilov & Hutter, 2019) claimed that $\mu/\sqrt{v} \approx 1$ in steady-state training, empirical findings show that the update is rather $\mu/\sqrt{v} \approx 0.2$, and the same for Muon with LR-adjustment to match the AdamW update (Liu et al., 2025). Therefore these optimizers apply parameter updates of RMS $0.2\,\eta(t)$, independent of weight magnitudes. The relative update then is $\|\Delta w\|_F/\|w\|_F = 0.2\,\eta(t)/\sigma_w = 0.2\eta(t)$ since for noNorm $\sigma_w \approx 1$ throughout training, and relative weight updates directly correspond to the chosen LR schedule.

We verify by comparing relative weight updates to the LR at two points during training from Fig. 24. At step 1000 (just past warmup, $\eta \approx 5{\times}10^{-2}$), $\sigma_w \approx 1.08$, predicting $0.2 \cdot 0.05/1.08 \approx 9.3{\times}10^{-3}$; observed is $8.2{\times}10^{-3}$. At step 95 400 (end of cosine decay, $\eta = 10^{-4}$), $\sigma_w \approx 1.10$, predicting $1.82{\times}10^{-5}$; observed is $1.6{\times}10^{-5}$. Both predictions are $\approx 12\%$ off from observed values.

We find that the cause is that Muon's Newton–Schulz coefficients do not produce a perfectly orthogonal update with only 5 iterations: with 5 NS steps, singular values have mean ≈ 0.90 and std ≈ 0.18, so the true per-entry update is $\approx 0.18\,\eta$ instead of $0.2\,\eta$ and the tighter envelope is $\|\Delta w\|_F/\|w\|_F \approx 0.18\,\eta(t)/\sigma_w$. This now matches noNorm to within a few percent.

E.1 Determining the optimal clip parameter c

In order to find suitable clip parameters c, we first analyze the evolution of magnitudes in a numerical simulation of weight updates. For controlled training dynamics, our goal is an approximate preservation of weight magnitudes throughout training such that the effective relative update steps adhere to the learning rate schedule. Figure 6 shows that clip parameters between 1.5 and 2.2 lead to magnitudes close to one, with expected deviations somewhat depending on specific step sizes resp. learning rate schedules. Starting at this predetermined candidate range, we identified in sweeps of small training runs that values around 2.1 perform slightly better. In addition, $c = 2.1$ barely affects a standard normal distribution, and allows us to use the same c for both weights and activations. Table 10 confirms this value results in best loss after quantization as well. Table 11 shows that overly large values for weight clipping impair training performance; Fig. 12 shows large activation clipping thresholds result in much faster growing activations. As seen in Figure 6, the type of constrained optimization (clipping with or without gradient projection) does not significantly influence the evolution of magnitudes. For this, Figure 8 shows an impact on post-training weight histograms.

E.2 Impact of weight decay on weight histograms

In their variant of the AdamW optimizer, Loshchilov & Hutter (2019) decouple weight decay regularization from the optimization steps during neural network training. Weight decay is conventionally expected to reduce the magnitude of the weights. In our architecture (1), taking control of the weight decay factor λ allows one to not just influence the spread but also the shape of the weight

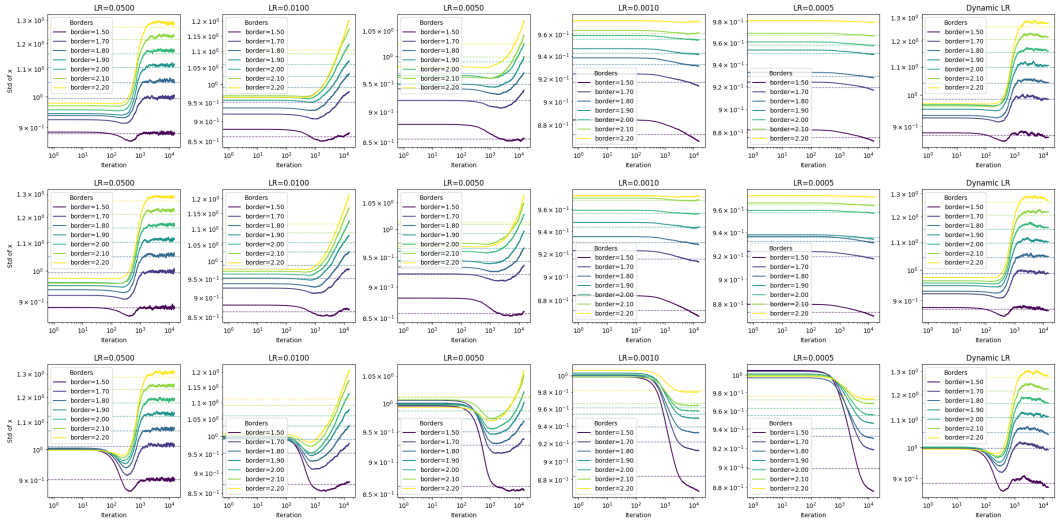


Figure 6: Evolution of magnitudes in a random walk starting at a point out of the standard normal distribution, in numerical simulations with different "learning" rates, and a dynamic schedule. Steps are simulated as Gaussian random walks with variances according to the learning rates. Linear warmup happens up to step 750, and for the dynamic schedule is followed by cosine warmdown from 0.05 to zero. From top to bottom: Clipping with gradient projection (as derived in Sec. 3), forced value clipping only, and gradient projection only without value clipping.

distribution. Weight decay pulls mass away from the clamp boundary $\pm c_w$: at low decay weights pile against it (bimodal), at high decay the distribution is close to a Gaussian around zero (Figs. 7 and 9). With typical weight decay factors, the weight distribution after training stays close to a Gaussian. Continually decreasing the weight decay factor, we observe a flattening of the distribution with growing peaks at the weight interval boundaries ("bath tub distribution"). Further decreasing the weight decay factor even below zero, the weights may grow towards the boundary of the clipping interval, resulting in a polarization of the weight distribution as shown in Fig. 7 (in the negative case we switch to L1 regularization for uniform impact). Training remains stable throughout, and the resulting performance is surprisingly similar. Bounded weights are already amenable to quantization, and polarization may support more aggressive quantization, for example binary $w_{ij} = \pm 1$.

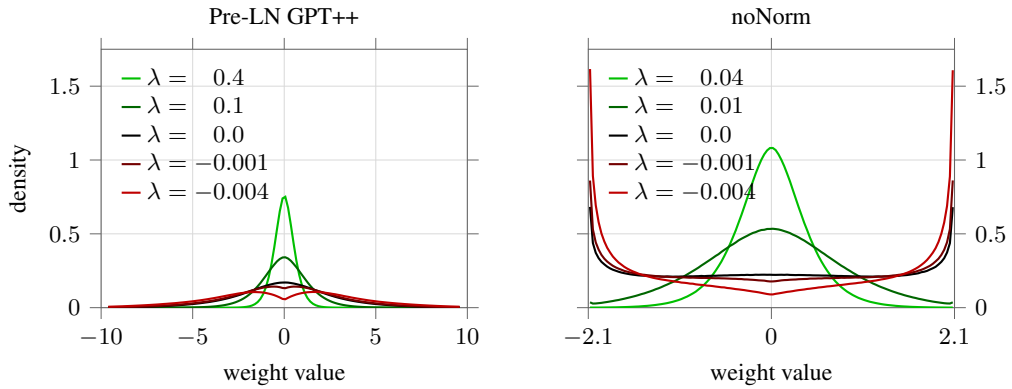


Figure 7: Post-training weight histograms for varying weight decay λ in Pre-LN GPT++ (left) and noNorm (right). Positive λ shrinks weights toward zero, producing a peaked distribution; negative λ pushes weights toward the clipping boundary, polarizing the distribution. The change in distribution for the noNorm architecture between $\lambda = 0.01$, 0, and -0.001 does not have a large impact on the resulting validation loss.

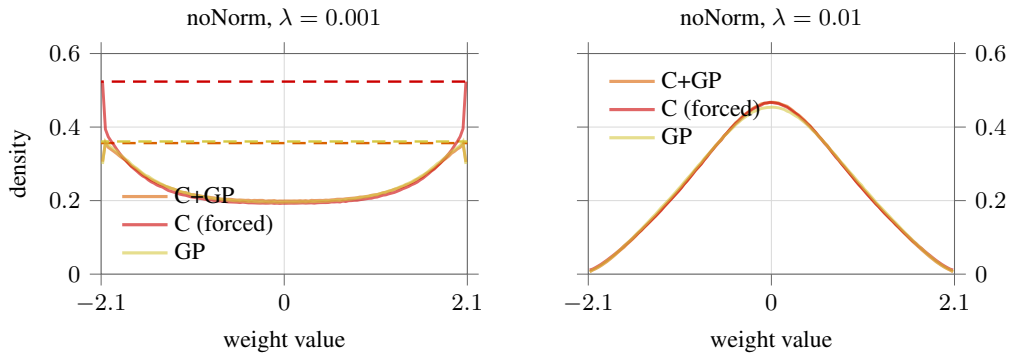


Figure 8: Post-training weight histograms of different clipping variants for solving the constrained optimization problem, with weight decay $\lambda = 0.001$ (left) and $\lambda = 0.01$ (right): Clipping with gradient projection (C+GP), forced value clipping only (C forced), and gradient projection only, no value forcing (GP). Dashed horizontal lines in the left panel mark the peak density of each curve, showing that gradient projection practically increases weight distribution smoothness.

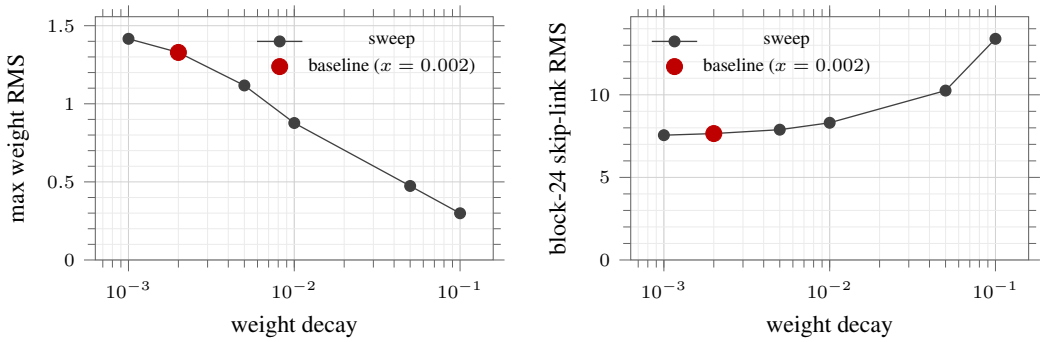


Figure 9: Final-checkpoint statistics vs. weight decay. Max weight RMS shrinks monotonically (left); deepest-block skip activation RMS grows because `lso` scales compensate (right).

F Ablations

We start from the noNorm 150M configuration (Table 6) and vary one option at a time, grouping related sweeps and reporting validation loss along with the change relative to the local baseline.

F.1 Run-to-run variance and significance threshold

We first characterize run-to-run standard deviation to set a significance threshold for the ablations (Table 7). Training nondeterminism alone (same initialization seed, five runs) gives $\sigma_{\text{training}} = 0.0007$; varying the initialization seed gives $\sigma_{\text{seed}} = 0.0018$, just over twice as high. All ablations share the same initialization seed and RNG state, so σ_{training} is the relevant scale, but for a safety margin we use σ_{seed}. The baseline validation loss used throughout is the 5-seed mean $\mu = 2.9111$, and we treat $|\Delta| < 2\sigma_{\text{seed}} = 0.0036$ as within noise. Almost all ablations we report below exceed this threshold, so a single run per ablation suffices.

Table 7: Baseline `noNorm-configurable` run-to-run mean μ and standard deviation σ at 150M/10B. Top block: cudnn nondeterminism only (same init seed, four extra runs). Bottom block: init-seed variance (distinct prime seeds). Lower validation loss is better.

Setting	$\mu \downarrow$	$\sigma \downarrow$	n
cudnn nondeterminism (same init seed)	2.9113	0.0007	5
init-seed variance (distinct seeds)	2.9111	0.0018	5

F.2 NOTATION

Shorthand used throughout this section: `onesided` = KKT-clipping (inward-pointing gradients preserved, Section 3.2); `hard` = standard clipping (gradients zeroed outside $[-c, c]$); `pclip` = post-step weight clipping; `div_lidx` = depth-dependent residual rescaling $1/\sqrt{\ell+1}$ (Section F, main text §Section 3.3); `lso` = learnable residual output scale ($s = e^{p-1.0}$, p learned); `lsi` = learnable residual input scale; `sq` = learnable per-head query scale; `emb_fwn` = Forced Weight Normalization applied to the embedding matrix.

F.3 OVERVIEW OF FINDINGS

Below results motivate the final noNorm configuration used in the 600M model in Table 6: we keep `div_lidx`, `sq` and `onesided` clamping for both weights and activations; for other options improvements are marginal.

- **Activation clamping** (Section F.4). Clamping of activations at either the input or output of residual blocks, or after the skip addition is required: removing it causes divergence. The best result is with residual output clamping only. Clamping at additional locations (input, skip, or all three) only hurts loss. Different clamp functions result in similar final loss, but `onesided` keeps skip-link activations smaller. Surprisingly, hard and KKT-based clipping of activations perform similarly even after quantization.
- **Weight clamping** (Section F.5). `tanh` is much worse than either `onesided` (KKT-clipping) or `hard` (standard clipping); the latter two give equal loss at BF16 but KKT clipping is better after quantization. Tight ($c_w = 1.0$) and very loose ($c_w = 10.0$) thresholds both increase the loss compared to $c_w = 2.1$. Too loose weight clamping leads to diverging effective learning rates.
- **Architecture** (Section F.6)
  - Depth-dependent residual scaling. Dividing the residual-stream input by $\sqrt{\ell+1}$ at every subblock improves loss.
  - Output scale (`lso`). Init values ≤ 0.01 all give the same loss, but initialization too large at ≥ 0.05 hurts loss. The exponential reparametrization clearly beats `linear` and `sigmoid`. Sharing a single α globally improves loss in isolation, but the gain disappears once `div_lidx` is enabled, so we do not use it for the large-model runs.
  - Query scale (`sq`). A learnable per-head query scale gives a small improvement.
- **Other knobs.** Learnable input scale `lsi` does not help; GLU with SiLU is not better than the 2-layer MLP with ReLU^2 baseline; applying FWN to embeddings doesn't make a difference. Learnable KV sink tokens improve loss slightly independent of model, but the improvement saturates at 64 tokens. Weight decay improves loss for small models (Table 11): $\lambda = 0.01$ improves loss over the $\lambda = 0.002$ baseline; for larger models the GPT-matched weight decay value works well (Equation (7)).
- **Scaling** At 150M parameters and 10B tokens, noNorm is worse than GPT++, but at 600M parameters and 100B tokens noNorm achieves better loss. Similarly, after scaling from 24 to 128 blocks noNorm overtakes GPT++.

Table 8: Architecture comparison at 150M/10B: validation loss for various models.

Setting	Val. loss ↓
GPT++	2.8759
FWN noLN	2.9086
nGPT	2.9567
noNorm ablation baseline (wdec 0.002)	2.9111
noNorm (wdec 0.01, div_lidx)	2.8515
noNorm (wdec 0.002, div_lidx)	2.8734

F.4 ACTIVATION CLIPPING (TABLE 9)

Clamp function: Disabling clamping results in divergence. With the same threshold of 2.1, final loss is almost identical for `onesided` and `hard`, but `hard` increases the skip-link RMS (Fig. 10 and Fig. 13). For activations, `tanh` clamping works equally well too. We scale the standard `tanh` function so it bounds at $c = 2.1$ instead of $c = 1.0$ as below. `tanh_lin` is a linearized version of `tanh` which is exactly linear up to $|x| = 0.7c$, then smoothly approaches $\pm c$. It is continuous at the junction points. For $c > 0$, let $a = 0.7c$. Note that c is fixed, not learnable.

$$\tanh_{\text{clip}}(x; c) = c \tanh\left(\frac{x}{c}\right), \tag{12}$$

$$\tanh_{\text{lin}}(x; c) = \begin{cases} x, & |x| \le a, \\ \operatorname{sgn}(x)\left[a + (c-a)\tanh\left(\dfrac{|x|-a}{c-a}\right)\right], & |x| > a. \end{cases} \tag{13}$$

Clamp location: removing clamping makes training diverge after ∼4k steps. Clamping only at the output of residual branches is best; additional clamping at the input is worse. Clamping on the skip link interferes with activations and gradients on the skip path and hurts loss.
Threshold: c_a values in $[1, 10]$ result in similar loss, however high thresholds significantly affect activation growth on the skip link as shown by Fig. 12. Fig. 16 shows how skip-link activation RMS at the end of the network varies depending on the activation clipping threshold. For the ablations using clamping on the skip link, clamping functions on the skip link use $c = 7$ in order to allow a number of residuals to accumulate freely.

Table 9: Activation-clamp ablations at 150M/10B. Deltas are measured relative to the local baseline within each block. Lower validation loss is better.

Study	Setting	Val. loss ↓	Δ
act clamp function	*baseline (onesided)*	2.9111	0.0000
act clamp function	hard	2.9105	-0.0006
act clamp function	tanh	2.9119	+0.0009
act clamp function	tanh_lin	2.9109	-0.0002
act clamp location	*baseline (out only)*	2.9111	0.0000
act clamp location	none	NaN	—
act clamp location	in only	2.9188	+0.0077
act clamp location	skip only	2.9153	+0.0042
act clamp location	in+out	2.9497	+0.0386
act clamp location	in+skip	2.9243	+0.0132
act clamp location	out+skip	2.9142	+0.0031
act clamp location	in+out+skip	2.9473	+0.0362
act clamp threshold	*baseline (2.1)*	2.9111	0.0000
act clamp threshold	1.0	2.9158	+0.0048
act clamp threshold	2.0	2.9118	+0.0008
act clamp threshold	2.3	2.9109	-0.0002
act clamp threshold	3.0	2.9097	-0.0014
act clamp threshold	10.0	2.9119	+0.0008

Table 10: Effect of weight and activation clipping method on loss when quantizing weights and/or activations, at 150M/10B. Deltas are relative to the baseline (lower is better).

Setting	BF16	W4 Δ	A4 Δ	W4A4 Δ
noNorm baseline	2.9111	+0.025	+0.264	+0.308
Weight clipping off	2.9297	+1.874	+0.512	+7.589
Activation clipping off	diverged	—	—	—
Hard activation clipping	2.9111	+0.026	+0.258	+0.304
Hard weight clipping	2.9108	+0.030	+0.319	+0.358
Hard activation and weight clipping	2.9084	+0.030	+0.319	+0.370

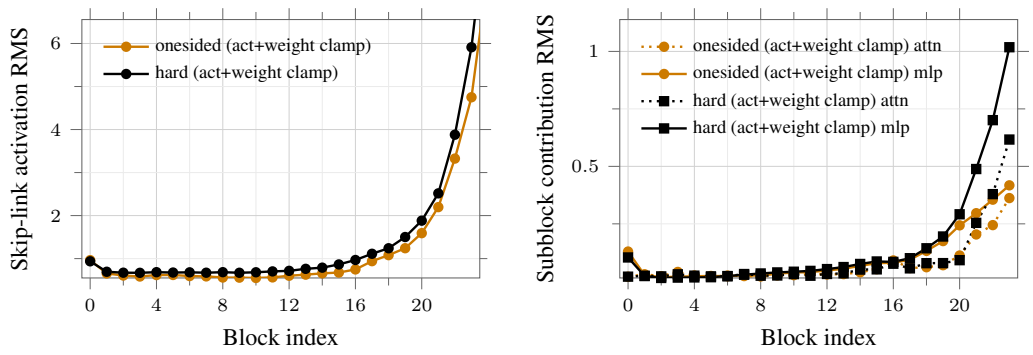


Figure 10: Activation clamp function ablation (Table 9): equal validation loss but `onesided` (our KKT-based clipping function) results in slower activation growth on the skip link relative to `hard`.

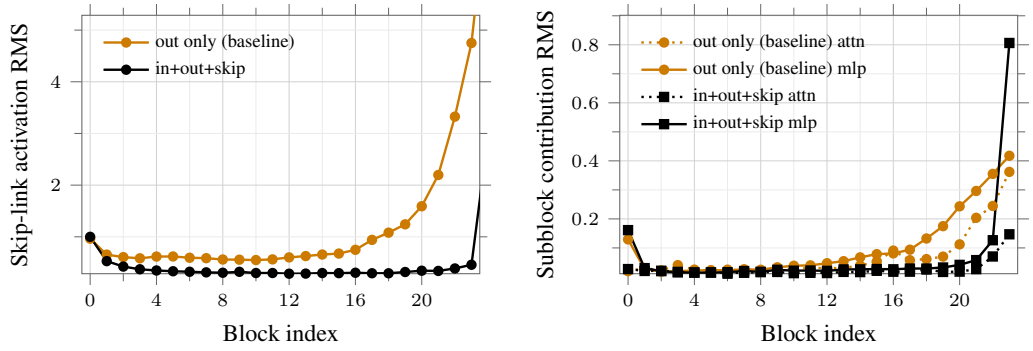


Figure 11: Activation clamp location ablation (Table 9): over-clamping (in+out+skip) results in a very high contribution from the last blocks and increases validation loss

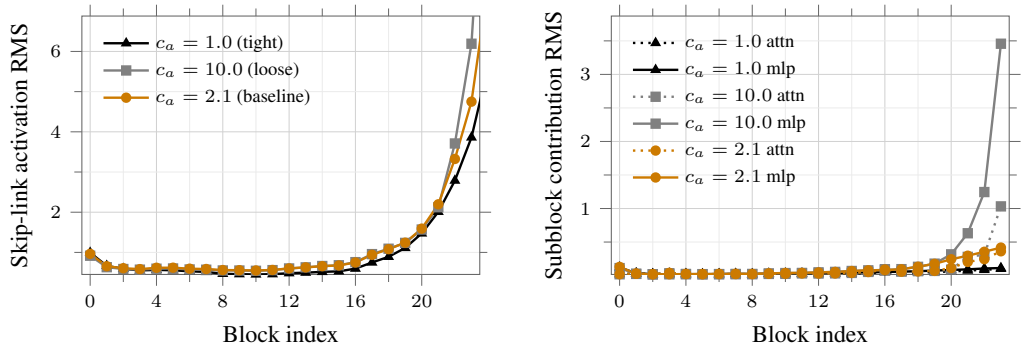


Figure 12: Activation clamp threshold ablation (Table 9): skip-link RMS varies 2.3× across a 10× range of c_a, even as differences in validation loss are negligible. However with high clipping thresholds, quantization is more difficult.

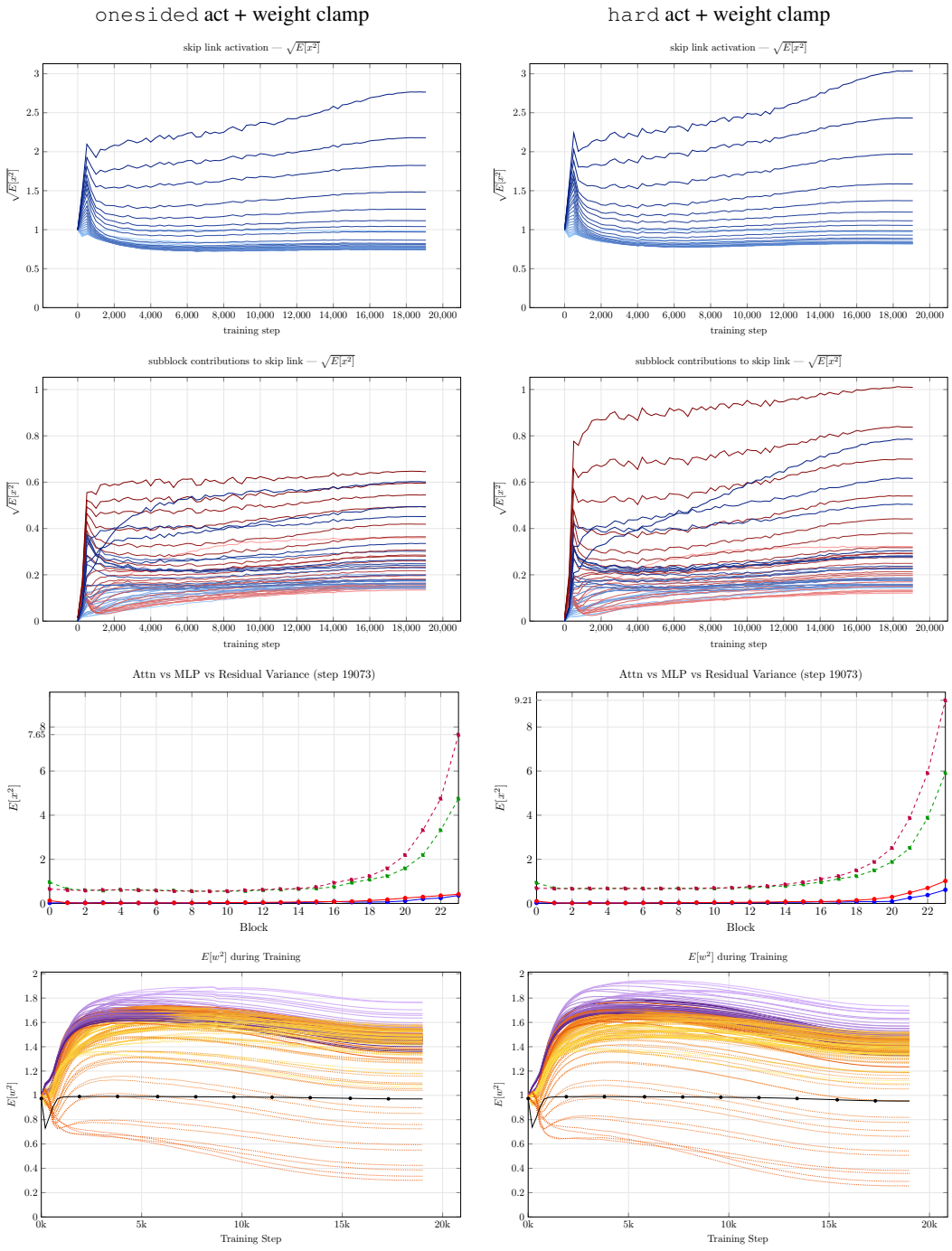


Figure 13: Training-time evolution of activations (top three rows) and weights (bottom row): `onesided` (left) vs. `hard` (right) act+weight clamping. `hard` produces higher skip-link RMS and lower weight variance throughout training.

F.5 WEIGHT CLIPPING (TABLE 11)

Clamp function: `tanh` is substantially worse (Δ=+0.050); `onesided` and `hard` give equal loss, and switching to `hard` weight clipping doesn't modify activation RMS as shown by Fig. 15. However it results in worse post-quantization loss (Table 10) not just when quantizing weights, but also when quantizing only activations.
Threshold: both extremes hurt loss: tight clamping piles weights against the boundary, loose clamping results in widely varying weight RMS throughout the model (Fig. 16) as well as outlier weights which make quantization difficult (Table 10). This can be explained by per-parameter analysis, where we find some weights become very large without bounding, with individual outlier weights exceeding 40. Fig. 14 and Fig. 23 additionally demonstrate that different weight tensors vary widely in RMS, no longer equalizing learning rates thus making training less controlled.

Table 11: Weight-control and regularization ablations at 150M/10B. Deltas are relative to the baseline (lower is better).

Study	Setting	Val. loss ↓	Δ
weight clamp function	*baseline (onesided)*	2.9111	0.0000
weight clamp function	none (pclip=False)	2.9291	+0.0180
weight clamp function	hard	2.9104	-0.0007
weight clamp function	tanh	2.9613	+0.0502
weight clamp threshold	*baseline (2.1)*	2.9111	0.0000
weight clamp threshold	1.0	2.9322	+0.0212
weight clamp threshold	2.0	2.9128	+0.0017
weight clamp threshold	2.5	2.9144	+0.0034
weight clamp threshold	3.0	2.9179	+0.0068
weight clamp threshold	5.0	2.9241	+0.0131
weight clamp threshold	10.0	2.9275	+0.0164
weight decay	*baseline (0.002)*	2.9111	0.0000
weight decay	0.1	2.9511	+0.0400
weight decay	0.05	2.8956	-0.0154
weight decay	0.01	**2.8856**	-0.0254
weight decay	0.005	2.8978	-0.0132
weight decay	0.001	2.9160	+0.0049

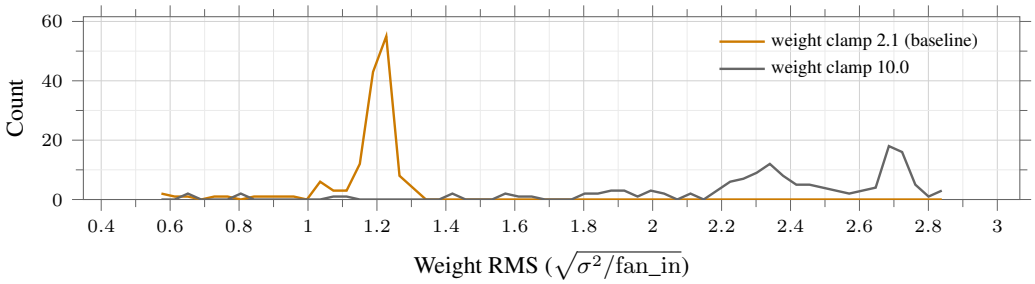


Figure 14: Weight-clamp threshold ablation (Table 11): weight clamping to c_w=2.1 results in similar RMS for all weight matrices. Setting a high threshold (c_w=10.0) results in a long weight tail. Per-parameter breakdown in Fig. 23.

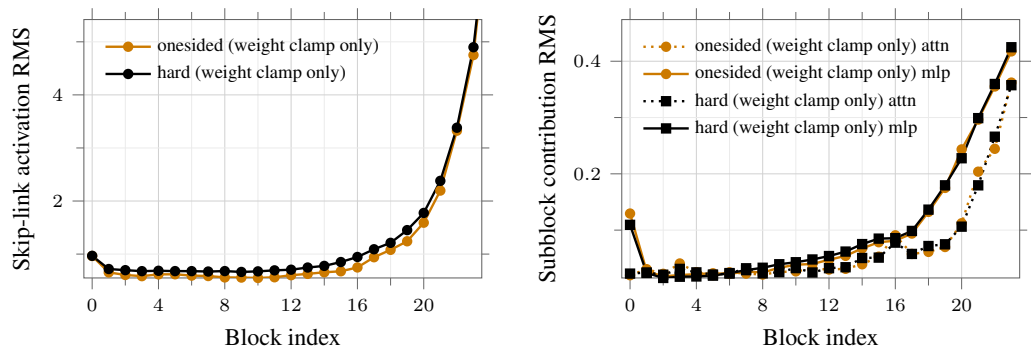
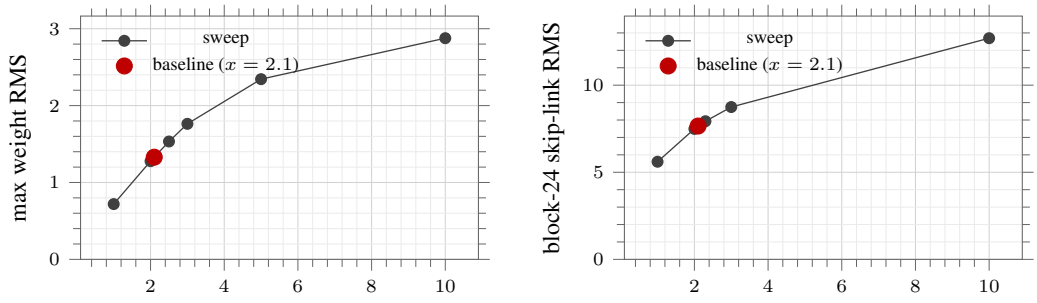


Figure 15: Weight-clamp function ablation (Table 11): with activation clamping held at `onesided`, switching the weight clamp to `hard` leaves the activation profile unchanged.



(a) Weight clamp threshold c_w.

(b) Activation clamp threshold c_a.

Figure 16: Clamp threshold sweeps and impact on weight and activation magnitudes (Tables 9 and 11). **(a)** Weight clamp: tighter bounds result in lower RMS. c_w=10.0 gives both high maximum RMS and a much larger spread (Fig. 14). **(b)** Activation clamp: 10× range of c_a results in a ∼2.3× increase in skip-link RMS. Also see Fig. 12 for a visualization of activation RMS across model depth.

F.6 Optimizer (Table 12).

Muon outperforms AdamW for both noNorm and GPT++. Loss curves are very similar (but lower for Muon). The LR warmup schedule has negligible effect, training works well without warmup. Using other learning rates for 2D weight matrices and 1D parameters (α scales) isn't required; too high learning rates for α can cause instability.

Table 12: Optimizer and learning rate ablations at 150M/10B. 1D parameters are `lso`, `sq`); baseline is 5e-2 for all parameters. Deltas are relative to the baseline (lower is better).

Study	Setting	Val. loss ↓	Δ
optimizer	*noNorm, muon (baseline)*	2.9111	0.0000
optimizer	noNorm, adamw	2.9320	+0.0209
optimizer	gptpp, muon (baseline)	**2.8759**	-0.0351
optimizer	gptpp, adamw	2.8913	-0.0198
lr schedule	*baseline (5e-2, 715 warmup steps)*	2.9111	0.0000
lr schedule	no warmup (no warmup)	2.9120	+0.0010
lr schedule	2D LR=5e-2, 1D LR=1e-3	2.9112	+0.0001
lr schedule	2D LR=5e-2, 1D LR=1e-2	2.9121	+0.0010
lr schedule	2D LR=5e-2, 1D LR=1e-1	5.4385	+2.5275

F.7 Architecture (Tables 10 and 14).

`div_lidx` significantly improves loss. KV sink tokens (Sun et al., 2024) improve loss slightly; benefit saturates by 64 tokens. For the learned output scale (Table 13), init values ≤ 0.01 result in the same loss; the exponential reparameterization clearly beats linear and sigmoid. A single global α shared across all blocks improves loss in isolation, but the gain disappears once `div_lidx` is enabled, so we do not use it. Adding a pre-normalization RMSNorm layer back into noNorm does not improve loss.

Table 13: Learned output scale (`lso`) ablations at 150M/10B: initialization value and reparameterization function. Deltas are relative to the baseline (lower is better).

Study	Setting	Val. loss ↓	Δ
lso init	*baseline (0.01)*	2.9111	0.0000
lso init	0.001	2.9111	+0.0000
lso init	0.05	2.9249	+0.0138
lso init	0.1	2.9300	+0.0189
lso reparametrization	*baseline (exp)*	2.9111	0.0000
lso reparametrization	linear	2.9332	+0.0221
lso reparametrization	sigmoid	2.9315	+0.0204

F.8 Depth (Table 15 and Figs. 17 and 18).

Both models improve substantially with depth ($\Delta\approx-0.23$ at B=128); loss curves are very similar for deep models and noNorm achieves lower loss at both depths. GPT++ skip-link RMS grows from $\sim$140 at B=24 to $\sim$230 at B=128, with the deepest blocks contributing disproportionately; noNorm stays within $\sim$17 and $\sim$55 respectively.

Table 14: Architectural ablations at 150M/10B. Normalization block: embedding FWN ablation and comparison with pre-LN and GPT++. Deltas are relative to the baseline (lower is better).

Study	Setting	Val. loss ↓	Δ
kv sink tokens	*baseline (0)*	2.9111	0.0000
kv sink tokens	64	2.8994	-0.0117
kv sink tokens	256	**2.8976**	-0.0135
div_lidx	*baseline (False)*	2.9111	0.0000
div_lidx	True	**2.8700**	-0.0411
GLU	*baseline (False)*	2.9111	0.0000
GLU	True	2.9200	+0.0087
normalization	*baseline (emb-FWN, no pre-LN)*	2.9111	0.0000
normalization	no emb FWN	2.9160	+0.0049
normalization	noNorm + pre-LN	2.8967	-0.0144
normalization	noNorm, wdec 0.01, sq 1.0	**2.8415**	-0.0696
normalization	noNorm, wdec 0.01, sq 1.0, pre-LN	2.8780	-0.0331
normalization	GPT++	2.8759	-0.0352
learnable scales	*baseline (none)*	2.9111	0.0000
learnable scales	sq + lsi (init=1.0)	2.9118	+0.0007
learnable scales	sq (init=1.0)	**2.9048**	-0.0063
learnable scales	lsi (init=1.0)	2.9122	+0.0011
learnable scales	sq (init=0.1)	2.9070	-0.0041
combination	*wdec=0.002, divlidx=False (baseline)*	2.9111	0.0000
combination	wdec 0.01, div_lidx=True	2.8523	-0.0588
combination	wdec 0.01, div_lidx=True, sq	2.8413	-0.0698
combination	wdec 0.01, div_lidx=True, sq, kv 64	**2.8368**	-0.0743

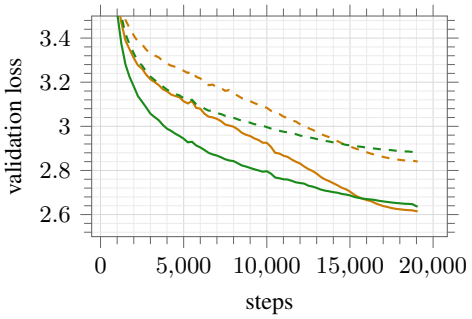


Table 15: Depth ablation at 150M/10B.

Setting	Val. loss ↓	Δ
noNorm (B=24)	2.8415	0.0000
GPT++ B=24	2.8759	+0.0347
noNorm, B=128	**2.6151**	-0.2261
GPT++ B=128	2.6382	-0.2031

Figure 17: Effect of depth: models with 24 and 128 blocks trained for 10B tokens

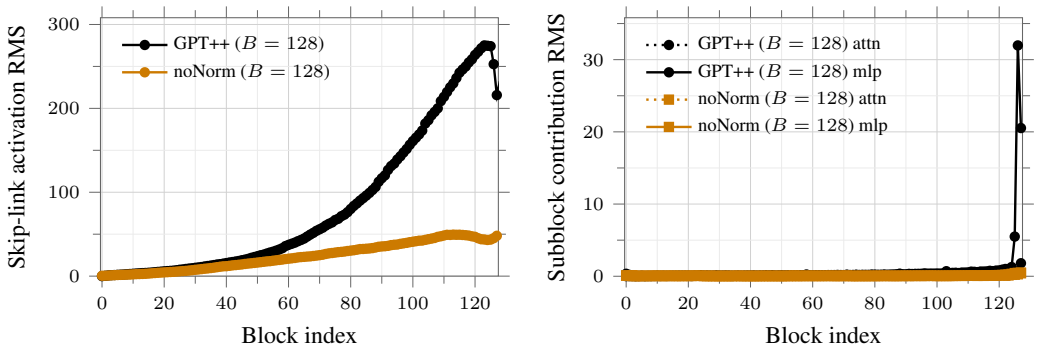


Figure 18: Activation profile at $B = 128$: deep MLP layers in GPT++ have huge outputs

G Analysis of Element-Wise Projection for Nested Clipping

The constrained optimization problem in Section 3.1 has two goals, bounding weight magnitudes and bounding residual activations. Enforcing box constraints that directly depend on free optimization variables is simple, and the implementation of weight bounding by a KKT-consistent projection of the gradients is straightforward, as stated in (5). Enforcing activation constraints during the optimization is the more interesting case, as the corresponding clipping operations are applied to a data-dependent mix of many variables at each point in training time. Clearly, there is no direct way of back-projecting optimization variables into the desired bounded domain. Rather, a combination of saturating the clipping function and steering the gradient back towards a more feasible region has to nudge training towards the desired domain over several time steps. This has been studied extensively in the form of anti-windup compensation in the face of control saturation (Teo & How, 2009). Formally, we account for the temporary excess of residual activation inputs $\mathbf{y}_\ell = \alpha_\ell \cdot \mathbf{F}_\ell(\mathbf{x}_\ell)$ by an additional constraint on the penetration depths:

$$\rho_h(\mathbf{y}) := \max\{0, |y_h| - c\}, \qquad h = 1, \dots, H. \tag{14}$$

Our goal is then to solve a secondary constrained optimization problem, that minimally modifies the update step such that the penetration depths do not increase. The challenge is to implement this projection in the general case. In the following, we show under which conditions local element-wise gradient projection still works well when the activation bounding is nested inside a deeper network, as is the case in a complex network architecture.

At each training time step, there are parameter- and time-dependent input activations $\phi(\theta)$ to the nested activation clipping operations $\mathbf{z} := \mathrm{clip}(\phi(\theta), c)$. The output and loss based on the clipped activations $\mathbf{z}$ is then computed as $f(\mathbf{z})$. Thus, the gradient $\boldsymbol{\gamma}$ in Section 3.1 becomes $\boldsymbol{\gamma} = \nabla_{\mathbf{z}} f$. The Jacobian $J := \partial\phi(\theta)/\partial\theta$ transforms it into parameter gradients and ultimately tentative update steps $\Delta\boldsymbol{\theta}_0 = -\eta J^T \boldsymbol{\gamma}$. The projected update step $\Delta\boldsymbol{\theta}$ limiting updates to the viable domain is obtained by minimizing its distance to the unconstrained update step while accounting for all active activation constraints, transformed to the space of parameters as $U_{h,\cdot} = \mathrm{sign}(y_h) J_{h,\cdot}$:

$$\mathcal{L}_\rho(\Delta\boldsymbol{\theta}, \boldsymbol{\nu}) = \tfrac{1}{2}\|\Delta\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_0\|_2^2 + \boldsymbol{\nu}^T U \Delta\boldsymbol{\theta}, \qquad \boldsymbol{\nu} \geq 0, \tag{15}$$

Note that the current penetration depth is irrelevant here, as we have already determined which constraints are active and know which half-spaces need to be excluded from the set of viable directions. By deriving the secondary Lagrangian $\mathcal{L}_\rho$, we arrive at the critical point for $\Delta\boldsymbol{\theta}^\star = \Delta\boldsymbol{\theta}_0 - U^T\boldsymbol{\nu}$. The multipliers are found by minimizing the quadratic form resulting from substitution:

$$\boldsymbol{\nu}^\star = \arg\min_{\boldsymbol{\nu}\geq 0} \ \tfrac{1}{2}\boldsymbol{\nu}^T U U^T \boldsymbol{\nu} - \boldsymbol{\nu}^T U \Delta\boldsymbol{\theta}_0. \tag{16}$$

Thus, the projected gradient is simple to obtain exactly element-wise (in $\boldsymbol{\nu}$, per clipped activation) whenever the product UU^T is diagonal, that is when the transformed constraint boundary normals remain orthogonal after transformation to the parameter space. This is trivially true for weight matrices right before activation clipping operations, where each row of the respective weight matrix uniquely influences one component of the clipped output activations. The same is also trivially true for vector-valued scaling coefficients in front of clipping operations. Past these trivial cases, orthogonality also approximately holds for constraints transformed via independent random matrices, which is the typical initial state of weight matrices. In the general case, element-wise gradient projection for activations may turn into more approximate steering. Note that UU^T restricted to the weight parameters that correspond to matrix multiplications right before respective activation clipping operations will remain diagonal, and thus the update restricted to these parameters will remain exact, still allowing the optimization to make progress towards viability along at least one route for each activation constraint in our architecture. Scalar-valued scaling coefficients are an interesting case in that a single value is affecting all downstream clipping operations simultaneously. If we were to solve for the projected scalar gradient exactly, the scalar would have to be decreased whenever a single constraint is violated, for penetration depth to decrease. In practice, we find the unified approximate steering to work well with scalar and vector-valued scaling coefficients. In the case of scalar coefficients, the effective update of element-wise activation gradient clipping is effectively the same as optimizing a vector-valued coefficient and then replacing it by its current average at the end of every update step.

H ADDITIONAL MODEL ANALYSIS

All figures in this section compare the 600M-parameter pre-LN GPT++ baseline against the noNorm model, both trained on 100B tokens. The subsections are ordered as follows: first training dynamics — skip-path variance across depth and throughout training (§H.1), showing that the depth-wise divergence is present from early in training; then weight scale and update statistics (§H.2); then activation statistics at the final checkpoint — heatmaps and distributions, and outliers (§H.3), which govern quantization behavior; then various other analyses (§H.4–H.4).

H.1 SKIP-PATH AND RESIDUAL RMS EVOLUTION OVER TRAINING

Figure 3a in the main text shows the final depth profile; Fig. 19 confirms the divergence is present *throughout training*. In pre-LN GPT++, only branch inputs are normalized, so branch outputs are unbounded. The skip link accumulates L branch contributions and grows with depth without bounds. noNorm does have bounded branch outputs due to activation clipping, resulting in much smaller activation growth with model depth, and no outliers on the skip path that may affect deeper blocks. Figure 20 tracks the RMS of each attention and MLP branch output (what each block adds to the skip path) over training.

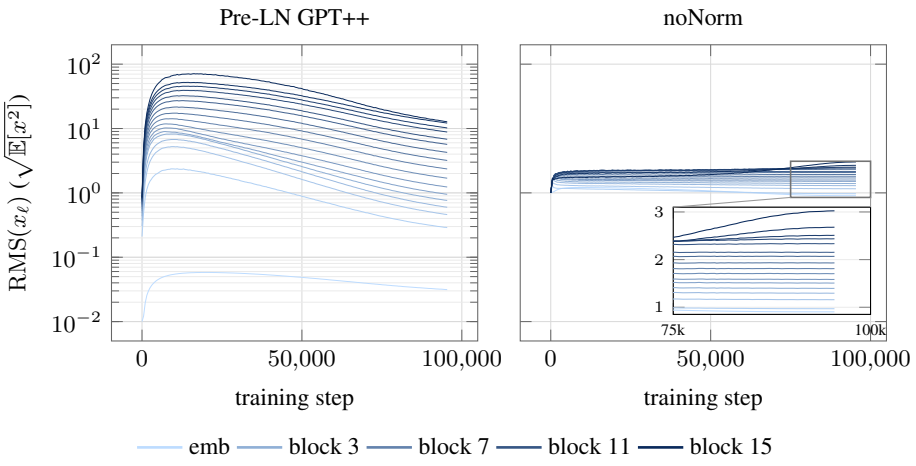


Figure 19: Skip-path RMS per block depth over training. Pre-LN: in deep blocks activation RMS increases to $\approx 100\times$ the embedding RMS in early training, then decreasing again to $\approx 13\times$. noNorm: skip-path RMS remains within $[0.5, 3]$ across blocks throughout training.

H.2 WEIGHT RMS AND RELATIVE UPDATES

For noNorm, per-parameter weight RMS stays close to 1 across all layers. Relaxing the weight clamp threshold lets individual parameters drift above this scale and also vary widely throughout the model (Figure 14). At $c_w = 10$, variances rise into the single-digit range across most layers and individual weights occasionally exceed 40 (Fig. 23). Bounding weight magnitudes keeps effective learning rates uniform across layers. Figure 24 shows the evolution over training of (i) per-parameter RMS evolution; and (ii) the per-parameter relative update $\|\Delta W\|_F / \|W\|_F$ (Figure 24).

Within a model, weight RMS varies depending on the layer that the weight is used in GPT++, i.e. attention or MLP layers, (Fig. 21) and the model depth (Fig. 22). This is not the case for noNorm however, where weight RMS values are approximately equal everywhere, which results in equal learning rates across modules and across depth.

Note: Pre-LN GPT++ initializes weights at $\sigma \approx 1/\sqrt{D}$ (output projections further scaled by $1/\sqrt{2L}$), while noNorm initializes at $\sigma \approx 1$ and through reparametrization (Section 3.2). We compare both relative to an RMS of 1.0 under the noNorm reparametrization, which preserves activation variance in expectation, by rescaling pre-norm weights by $\sqrt{D}$.

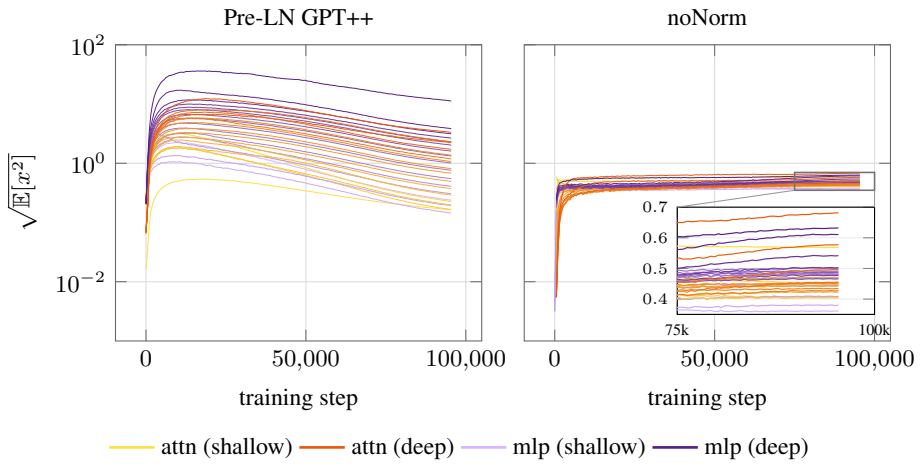


Figure 20: RMS of residuals of attention- and MLP-branch as a function of training step, one line per block each. Pre-LN (left): branch RMS spans orders of magnitude depending on block depth, and MLP branches have larger outputs than attention. Branch output magnitudes also vary widely throughout training. noNorm (right): attention and MLP branches of different depths all have similar magnitude outputs, and they remain much more stable throughout training.

Layer Type	Count	Min RMS	Mean RMS	Max RMS	Mean RMS $\cdot\sqrt{D}$
Attn Q	16	0.0292	0.0315	0.0342	1.2358
Attn K	16	0.0284	0.0311	0.0340	1.2189
Attn V	16	0.0279	0.0379	0.0455	1.4868
Attn Proj	16	0.0288	0.0352	0.0415	1.3790
MLP Up	16	0.0356	0.0371	0.0394	1.4525
MLP Down	16	0.0331	0.0374	0.0391	1.4674
Embedding	1	0.0351	0.0351	0.0351	1.3740
Other	1	0.0246	0.0246	0.0246	0.9634

(a) Pre-LN GPT++

Layer Type	Count	Min RMS	Mean RMS	Max RMS
Attn Q	16	1.0617	1.2054	1.2704
Attn K	16	1.0738	1.2116	1.2762
Attn V	16	1.1039	1.1422	1.1671
Attn Proj	16	1.0633	1.0902	1.1261
MLP Up	16	1.0713	1.0986	1.1419
MLP Down	16	1.0993	1.1102	1.1435
Embedding	1	1.1401	1.1401	1.1401
Other	1	0.9529	0.9529	0.9529

(b) noNorm

Figure 21: Final weight RMS grouped by layer type ($\sqrt{\mathbb{E}[w^2]}$ at the last training step). The pre-LN scaling of $\sqrt{D} \approx 39.2$ cancels the $1/\sqrt{D}$ initialization for comparison with noNorm; an RMS of 1.0 means activation variance is preserved in expectation.

Block	Count	Min RMS	Mean RMS	Max RMS	Mean RMS $\cdot\sqrt{D}$
embedding	1	0.0351	0.0351	0.0351	1.3740
block 0	6	0.0295	0.0326	0.0374	1.2791
block 1	6	0.0279	0.0337	0.0394	1.3198
block 2	6	0.0288	0.0334	0.0392	1.3072
block 3	6	0.0300	0.0335	0.0382	1.3137
block 4	6	0.0314	0.0336	0.0375	1.3167
block 5	6	0.0314	0.0341	0.0370	1.3376
block 6	6	0.0316	0.0345	0.0370	1.3526
block 7	6	0.0311	0.0351	0.0382	1.3745
block 8	6	0.0312	0.0355	0.0395	1.3926
block 9	6	0.0311	0.0359	0.0407	1.4084
block 10	6	0.0310	0.0361	0.0414	1.4144
block 11	6	0.0303	0.0365	0.0436	1.4312
block 12	6	0.0302	0.0361	0.0428	1.4166
block 13	6	0.0300	0.0370	0.0455	1.4519
block 14	6	0.0284	0.0358	0.0452	1.4027
block 15	6	0.0305	0.0371	0.0450	1.4554
lm_head	1	0.0246	0.0246	0.0246	0.9634

(a) Pre-LN GPT++

Block	Count	Min RMS	Mean RMS	Max RMS
embedding	1	1.1401	1.1401	1.1401
block 0	6	1.0617	1.1072	1.1435
block 1	6	1.0891	1.1383	1.1910
block 2	6	1.0752	1.1255	1.1829
block 3	6	1.0851	1.1267	1.1752
block 4	6	1.0906	1.1387	1.2105
block 5	6	1.0929	1.1481	1.2303
block 6	6	1.0939	1.1541	1.2446
block 7	6	1.1029	1.1504	1.2094
block 8	6	1.1027	1.1734	1.2762
block 9	6	1.0874	1.1670	1.2751
block 10	6	1.0967	1.1631	1.2505
block 11	6	1.0633	1.1557	1.2744
block 12	6	1.0645	1.1261	1.1805
block 13	6	1.0753	1.1375	1.1990
block 14	6	1.0747	1.1400	1.2128
block 15	6	1.0713	1.1372	1.2022
lm_head	1	0.9529	0.9529	0.9529

(b) noNorm

Figure 22: Final weight RMS per block — same $\sqrt{\mathbb{E}[w^2]}$ as Fig. 21, broken down by block index instead of layer type.

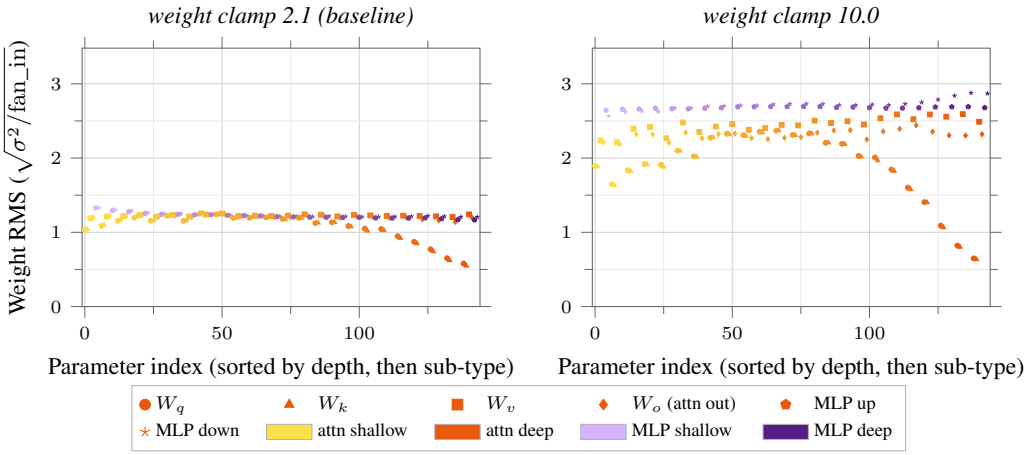
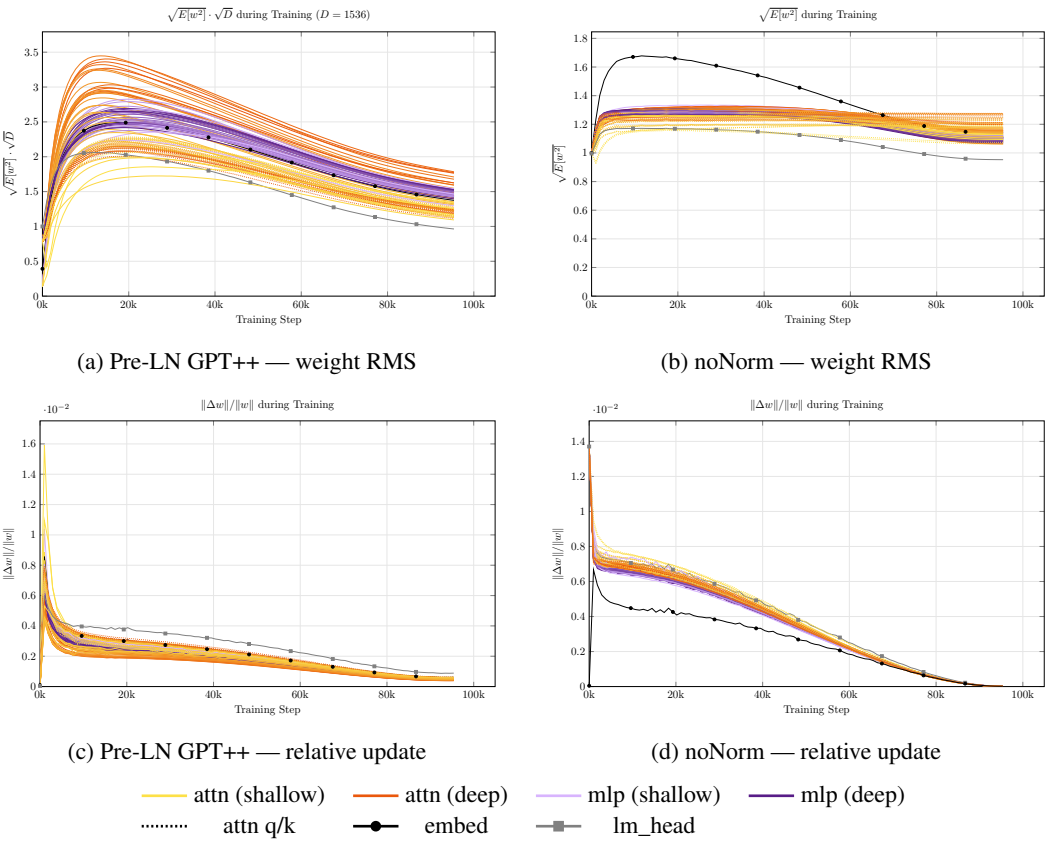


Figure 23: Per-parameter weight RMS at the final checkpoint, weight clamp $c_w = 2.1$ (baseline, left) vs. disabled $c_w = 10.0$ (right). Marker shape encodes sub-type; marker colour encodes block depth.



(a) Pre-LN GPT++ — weight RMS

(b) noNorm — weight RMS

(c) Pre-LN GPT++ — relative update

(d) noNorm — relative update

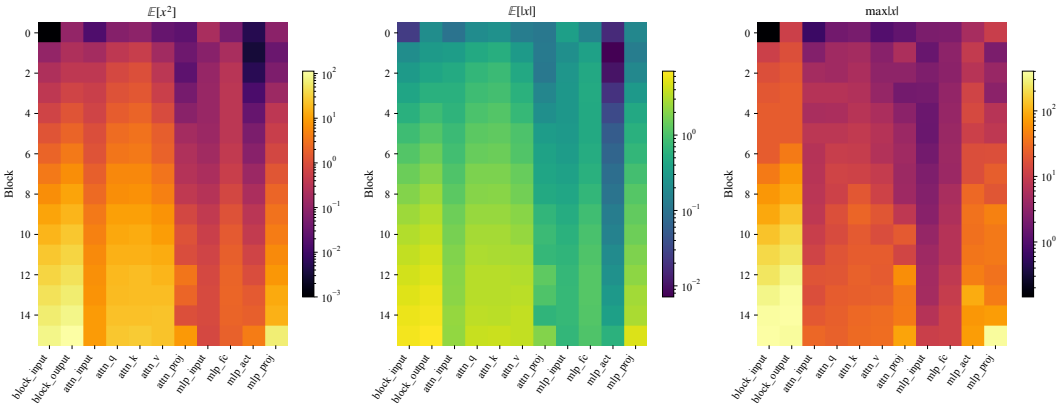
Figure 24: **Top:** per-parameter weight RMS over training (pre-LN rescaled by $\sqrt{D}$). Pre-LN grows then decays; noNorm stays near init throughout. **Bottom:** per-parameter relative update $\|\Delta w\|_F/\|w\|_F$. Pre-LN: updates vary across matrix types because σ_w differs ($1/\sqrt{2L}$ scaling on output projections) and weights drift during training. noNorm: all matrices receive similar relative updates, tracking the LR schedule.

H.3 Activation analysis

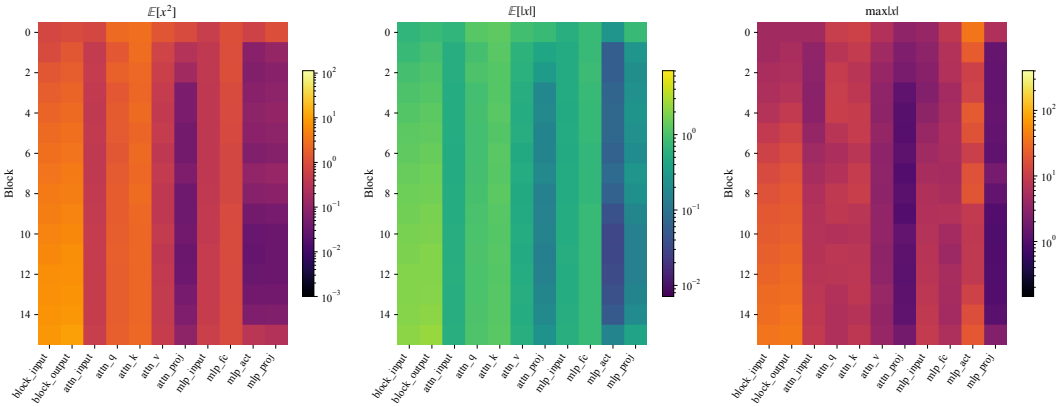
Quantization error scales with the range of the values being quantized: a layer whose activations span four orders of magnitude cannot share a single scale factor with a layer whose activations are tightly bounded. Figure 25 shows the residual stream and every attention/MLP intermediate activation. High dynamic range makes quantization more difficult. We pass a single prompt through the model and record activations at many locations, reporting uncentered variance ($\mathbb{E}[x^2]$) and mean absolute value ($\mathbb{E}|x|$) as well as $\max|x|$ at each location.

H.3.1 Per-block outlier-channel severity

Figure 3b in the main text shows only the relative ratio $\max_c / \text{mean}_c$; here we show absolute numbers for reference.



(a) Pre-LN GPT++

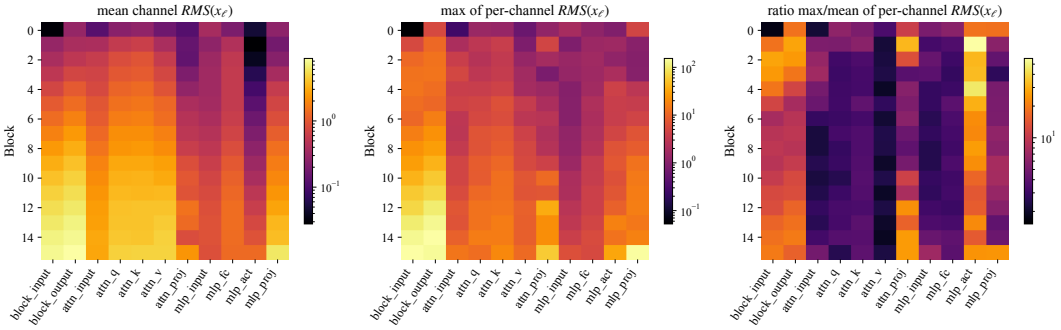


(b) noNorm

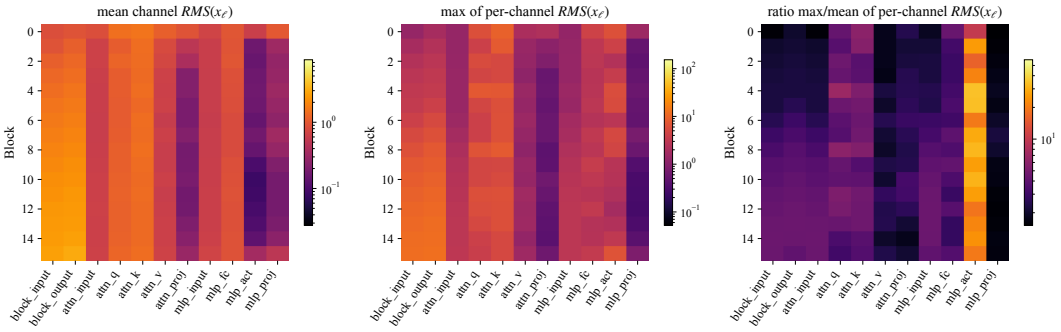
Figure 25: Activation statistics (uncentered variance, mean absolute, max absolute) at different locations inside each of the 16 blocks. Pre-LN $\mathbb{E}[x^2]$ ranges from $\approx 10^{-3}$ to $\approx 10^2$ and grows clearly from early to late blocks (top to bottom). noNorm stays within $[\approx 0.05,\ 10]$ at every depth.

H.3.2 Activation distributions

Range statistics ($\mathbb{E}[x^2]$, max) describe how wide a distribution is, but not whether mass is concentrated at the center or in the tails. Heavy tails (high kurtosis) are the shape signature of outliers, which dominate quantization error even when the mean variance is low. Skewness tells us whether the positive and negative tails are symmetric, a prerequisite for zero-point-centered quantizers. Figure 27 reports excess kurtosis $\frac{\mathbb{E}(x-\mu)^4}{\sigma^4} - 3$ and skewness $\frac{\mathbb{E}(x-\mu)^3}{\sigma^3}$ per (layer, location), pooled over tokens and channels.

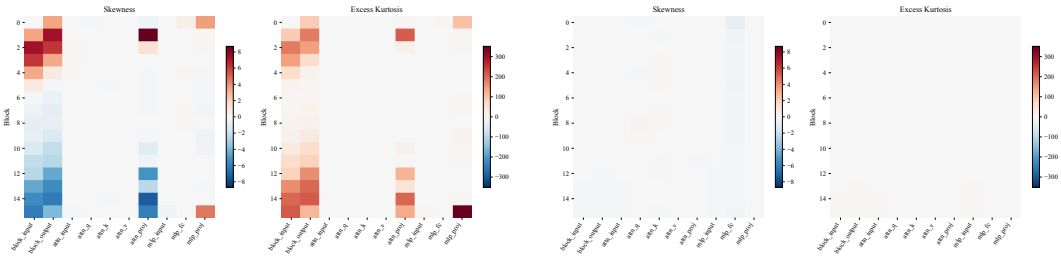


(a) Pre-LN GPT++



(b) noNorm

Figure 26: Per-block outlier-channel severity (rows = block, columns = activation location). Left to right: channel-mean $\mathrm{mean}_c = \sqrt{\mathbb{E}[x^2]}$, worst-channel $\mathrm{max}_c = \sqrt{\mathbb{E}[x^2]}$, and the ratio $\mathrm{max}_c\,/\mathrm{mean}_c$. Pre-LN GPT++ has many outlier-dominated cells across all three metrics; noNorm far less so. The `mlp_act` column is very high for both due to the ReLU^2 MLP activation.



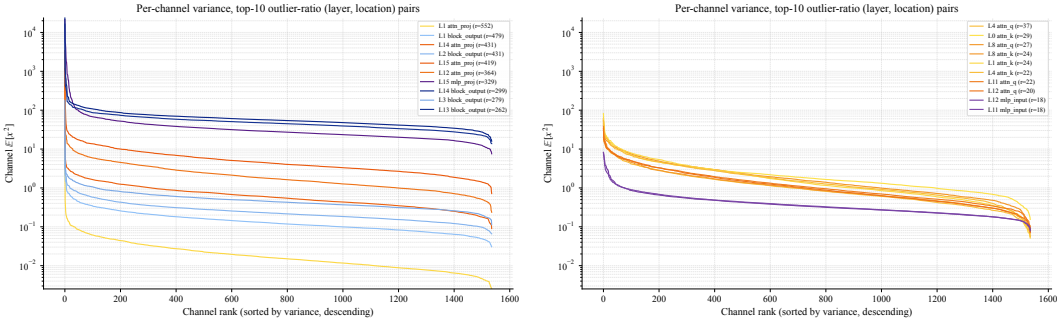
(a) Pre-LN GPT++

(b) noNorm

Figure 27: Skewness (left column of each subfigure) and excess kurtosis (right column) at different internal locations in each model. noNorm activations are much better behaved than GPT++.

H.3.3 OUTLIER CHANNELS

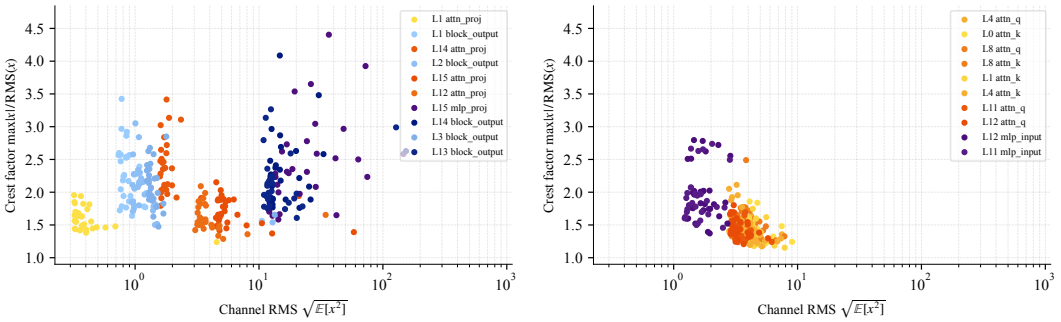
The main-text outlier heatmap (Fig. 3b) shows outlier channels can be an issue, where one channel has much larger activations than other channels. Figure 28 shows not just the mean and max but all per-channel variances, sorted in descending order for the top-10 (layer, location) pairs with highest outlier-channel ratio in each model. We note that even with outlier channels, if a channel's activations are always large, quantization with per-channel scaling would recover most of the quantization headroom. This is not the case for high dynamic range channels.



(a) Pre-LN GPT++ (b) noNorm

Figure 28: Per-channel variance at the top-10 (layer, location) pairs by outlier-channel ratio. Pre-LN: the worst locations have r up to ≈ 500, with very high peaks, indicating a single catastrophic channel. noNorm: r stays below ≈ 50 and the curves decay more gradually.

H.3.4 OUTLIERS: INDIVIDUAL ACTIVATIONS



(a) Pre-LN GPT++ (b) noNorm

Figure 29: Crest factor ($\max|x_c|/\mathrm{RMS}(x_c)$) for the top-30 channels with highest variance in each of the 10 internal activations with bad outlier-channel ratios (see Fig. 3b), excluding mlp-act. Pre-LN GPT++ has slightly higher crest factors, and far larger range of channel RMS, which makes quantization more difficult.

A high per-channel variance $\mathbb{E}[x_c^2]$ can arise in two ways: (1) sparse spike — most activations in channel c are small, but occasionally one activation takes a huge value; or (2) uniformly large — every activation in c is large. These cases have very different quantization implications: a sparse spike requires a wide dynamic range to represent accurately, whereas a uniformly large channel is well-handled by per-channel scaling. We distinguish them via the *crest factor* $\max|x_c|/\mathrm{RMS}(x_c)$: a crest factor near 1 indicates uniform magnitude; a large crest factor indicates a sparse spike, i.e. individual outlier activations which are very difficult to deal with when quantizing.

Figure 29 looks at the top 10 locations in the networks with high channel-outlier ratio (one channel has higher variance than others), in the assumption that these channels may also have high individual outliers (these are the locations that are bright in Fig. 3b). The plot shows the top 30 channels with highest variance in each of these 10 locations, with channel RMS on the x-axis and crest factor on

the y-axis. We exclude the mlp-act locations, as large activations there are solely due to the ReLU^2 activation, not due to training dynamics.

We find that pre-LN GPT++ channels have slightly higher crest factors than noNorm, but not dramatically so. The horizontal spread in the pre-LN GPT++ plot is caused by the later layers having much larger activations on average, which in turn is caused by the large growth of activations on the skip link (see Fig. 19). We note that literature looking at individual outliers reported that these are mostly a large-model phenomenon, e.g. in $> 8B$ parameter models, which may explain why these are not an issue even for GPT++ (Sun et al., 2024).

H.4 Effect of KV sink tokens

Both GPT++ and noNorm can benefit from learnable KV tokens that are internal to the attention module. In the literature there are different interpretations of these tokens, either as a temporary 'reasoning scratchpad', a 'memory/learnable prompt' token, a learnable softmax temperature, or as a way for the network to learn bias (Xiao et al., 2024; Fu et al., 2025; Darcet et al., 2024; Gu et al., 2025; Sun et al., 2024). Our ablations show that adding up to 64 KV tokens improve validation loss slightly, independent of the architecture (Table 14).

Sun et al. (2024) report that prepending learnable KV sink tokens to attention substantially reduces massive-activation outliers in pre-LN transformers. However, they report outliers are mostly a problem of very large LLMs and Section H.3.4 shows that outliers are not an issue at the model sizes we evaluated. For GPT++, our visualization of outlier channels in Fig. 3b shows that the tokens do not solve the outlier channel issue in GPT++. We did not find significant differences in outliers with the noNorm architecture by adding KV tokens. The noNorm design also clips outliers at outputs of residual blocks before they can affect the skip link activations, and avoids weight outliers by clipping, so fewer outliers are expected.

H.5 Learned scales on the residual branches

Figure 30 compares the per-block learnable-scale parameters of both models: for GPT++ it's the learnable scales at the input of the residual branches γ, for noNorm it is the scale at the end of the residual branch α. Both control how aggressively each branch contributes to the residual stream. In noNorm, α scales grow similarly to the activation RMS on the skip link, indicating that the relative scale of the residuals compared to the skip link is approximately constant throughout the network.

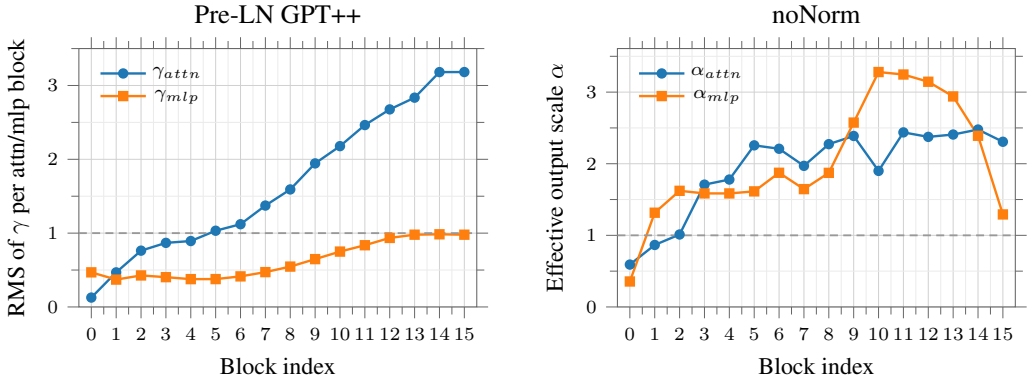


Figure 30: **Left (Pre-LN GPT++):** per-block RMS of the pre-branch RMSNorm gain (γ_{attn}, γ_{mlp}). **Right (noNorm):** effective per-block branch output scale (α_{attn}, α_{mlp}), applying Eq. (3) to α'.

I Quantization details

Post-training quantization with naive round-to-nearest (RTN), per-tensor symmetric scaling, no calibration data. Only 2D Linear weights with min-dim $\geq$ 128 are quantized; embeddings, the

unembedding, and 1D scales remain in BF16. Activations, where applicable, are quantized per-token. This is simple but strict PTQ; per-channel scaling, GPTQ, or SmoothQuant would improve results.

Per-token activation quantization means one INT-N scale per $(\text{batch}, \text{seq position})$, computed dynamically from $\max|x|$ across the channel dim. Fake-quant is applied only to the inputs of eligible `nn.Linear` layers (Q/K/V and output projections, MLP linears) via forward pre-hooks; outputs return to BF16. Quantization is applied before each linear matmul. Since RMSNorm statistics and residual accumulation are sensitive to quantization, those activations remain in bfloat16.

J BEYOND THE SCOPE OF THE MAIN PAPER

J.1 NON-NORMALIZED ATTENTION

Attention as introduced by Vaswani et al. (2017) applies the softmax function to query-key scores, thereby normalizing attention weights across the sequence dimension. This fixed-sum constraint removes one degree of freedom: one attention weight is determined by the others. Ran-Milo (2026) argues that attention sinks are induced by sequence-wise normalization and that non-normalized ReLU attention avoids this pathology. Wortsman et al. (2023) explores the same idea, replacing softmax with ReLU on query-key inner products scaled by the inverse sequence length.

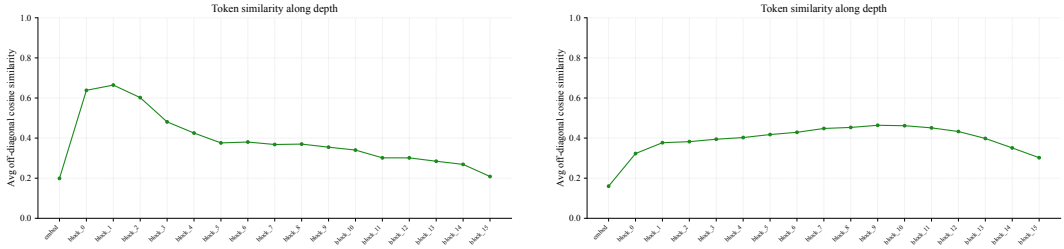
Initial experiments (150M parameters) show that replacing softmax with ReLU in self-attention yields nearly identical validation and training loss throughout pre-training. Scaling by the inverse context length, as suggested by Wortsman et al. (2023), had no measurable effect. However, these models generalize poorly beyond the training sequence length: loss degrades noticeably faster than with standard softmax attention. Addressing this is a promising direction for future work. We hypothesize this explains why softmax-free attention works well for vision models (Ran-Milo, 2026; Wortsman et al., 2023), which have a fixed context length.

J.2 QUANTIZING GRADIENTS: CONSISTENT VALUE RANGES

The embedding vectors are initialized to have unit variance. This matches u-μP (Blake et al., 2025). Their ABC-scaling is incompatible with weight tying; we find that a uniform $1/\sqrt{d_{\text{in}}}$ scaling also works for tied output matrices. To bring gradients within a controlled range, the scale factors of matrix multiplications must be adjusted. Specifically, scaling by $1/\sqrt{d_{\text{out}}}$ counteracts the output matrices in the backward pass, without changing the relative gradient weights across branches. Depending on the implementation, softmax gradients may be excessively small; this can be addressed by scaling by $\sqrt{\texttt{vocab_size}}$ and by $1/\sqrt{\texttt{micro_batch_size} \cdot \texttt{seq_len}}$ to normalize across gradient accumulation steps.

J.3 TOKEN REPRESENTATION DIVERSITY

A known failure mode in deep transformers is *representation collapse*: tokens converge to nearly the same hidden vector at some depth. Figure 31 shows mean off-diagonal cosine similarity between token representations per layer for a fixed prompt. GPT++ has high cosine similarity early in the model, and needs multiple blocks to reduce it. noNorm has more stable and lower cosine similarity throughout the model.



(a) Pre-LN GPT++ (b) noNorm

Figure 31: Mean off-diagonal cosine similarity of token representations at each layer.