KiteNorm: Variance Regularisation for Stable and Scalable Post-LN Transformers

Leon A. Trochelmann L.TROCHELMANN@PROTON.ME
ELLIS Institute Tübingen, Tübingen AI Center, Germany

Sajad Movahedi SAJAD.MOVAHEDI@TUE.ELLIS.EU
ELLIS Institute Tübingen, MPI-IS, Tübingen AI Center, Germany

Shiwei Liu SLIU@TUE.ELLIS.EU
ELLIS Institute Tübingen, MPI-IS, Tübingen AI Center, Germany

Antonio Orvieto ANTONIO@TUE.ELLIS.EU
ELLIS Institute Tübingen, MPI-IS, Tübingen AI Center, Germany

Abstract

Post-LayerNorm Transformers have seen limited practical adoption due to enduring difficulties in scaling them to large depths. While prior research has focused on stabilising Post-LayerNorm by improving conditioning at initialisation, stability often deteriorates when models are trained with large learning rates, forcing additional compromises. In the decoder-only settings we study, existing Post-LayerNorm methods fail to outperform strong Pre-LayerNorm baselines. We propose **KiteNorm**, a novel normalisation method designed to break this trend. KiteNorm achieves stability through residual scaling and a regularisation technique based on hidden-state variance. Beyond stabilisation, KiteNorm learns separate scales for the skip and residual branches in each sublayer, improving performance. Across decoder-only Transformers up to 1B parameters, KiteNorm remains stable throughout training and consistently outperforms leading baselines, with scaling laws favouring KiteNorm across depth, width, batch size, and training budget. Ablations show that each component is necessary for the full stability and performance gains. We offer theoretical insights into KiteNorm through a new perspective on the gradient vanishing problem in Post-LayerNorm, linking the underlying hidden-state expansion to rank collapse. Code is available at https://github.com/l-trochelmann/KiteNorm.

1. Introduction

Post-LayerNorm (post-norm) Transformers [24] have largely been replaced by pre-LayerNorm (pre-norm) Transformers [26] in large-scale Transformer implementations, owing to their improved gradient conditioning along the uninterrupted skip branch [28]. However, pre-norm exhibits persistent weaknesses, such as parameter inefficiency in deep layers [7, 23] and massive activations [22]. Residual scaling [18, 23] can mitigate these weaknesses, but it cannot fully eliminate them as the architecture necessarily piles up sublayer contributions on the skip branch.

This has motivated renewed interest in post-norm architectures with the goal of scaling them to a large number of layers. The core challenge is the fact that the Jacobian of post-norm acts on both the residual and skip branches during backpropagation, which leads to gradient vanishing as the scale of the activations and the number of layers rise [9, 15]. While prior work has analysed the dynamics of this phenomenon, the underlying cause remains poorly understood. We extend existing insights by

© L. A. Trochelmann, S. Movahedi, S. Liu & A. Orvieto.

formally proving the effect of hidden-state variance on gradient propagation, and by linking gradient vanishing to rank collapse [6, 16].

Recent works have made strides in improving the conditioning of post-norm Transformers by applying scaling factors to the residual branches [3, 10, 25]. While these methods improve optimisation conditions at initialisation, they can be less robust to high learning rates or less competitive at moderate depths. We postulate that methods that rely primarily on favourable initialisation conditions are ultimately prone to degraded performance or stability as large learning rates push models further away from initialisation. Resolving this trade-off is therefore important for improving the scalability and efficiency of Transformer training. Our contributions can be outlined as follows. We extend the theoretical analysis of gradient vanishing in post-norm Transformers and provide empirical evidence linking it to rank collapse. We propose KiteNorm, including hidden-state variance regularisation, and demonstrate improved stability and scaling up to 1B parameters. We further provide ablations showing that each component is essential.

2. Preliminaries

Appendix A.1 gives the full notation and rank-collapse metric. In the main text, it suffices to write $\boldsymbol{X}_l$ for a sequence of hidden states entering sublayer l, $\mathcal{F}_l$ for its residual branch, and $\boldsymbol{Z}_l = \boldsymbol{X}_l + \mathcal{F}_l(\boldsymbol{X}_l)$ for the hidden states after the residual addition. Define the normalisation of a vector $\boldsymbol{v} \in \mathbb{R}^d$ as follows: $\tilde{\boldsymbol{v}} = \frac{\boldsymbol{v}-\mu(\boldsymbol{v})\mathbb{1}}{\sigma(\boldsymbol{v})}$, where μ and σ denote the mean and standard deviation, respectively, and $\mathbb{1}$ is the size-d vector of ones. LayerNorm [1], as used in Transformers, can then be written as $\mathrm{LN}(\boldsymbol{X}) = \boldsymbol{\gamma} \cdot \tilde{\boldsymbol{X}} + \boldsymbol{\beta}$, where $\boldsymbol{X} \in \mathbb{R}^{S\times d}$ is a length-S sequence of tokens.

2.1. Gradient Propagation in Post-Norm Transformers

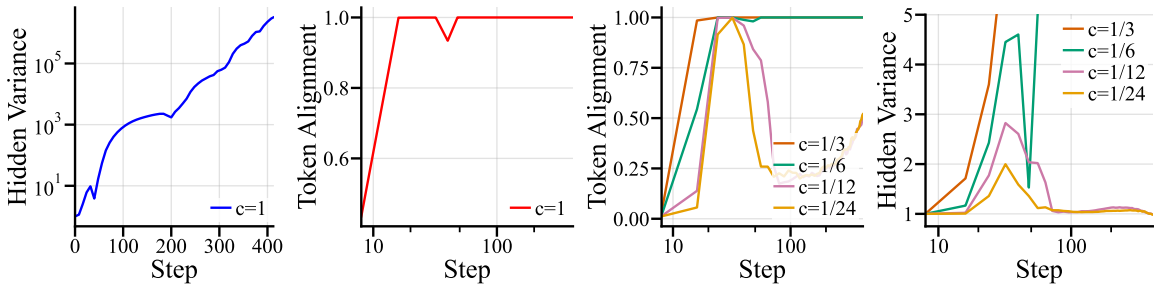
We now examine the backward pass of post-norm Transformers. Let LN_l denote the LayerNorm operation in sublayer l, and let $\boldsymbol{x}_{l+1} = \mathrm{LN}_l(\boldsymbol{x}_l + \mathcal{F}_l(\boldsymbol{x}_l)) = \mathrm{LN}_l(\boldsymbol{z}_l)$. Then

$$\frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l} = \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l}\frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{x}_l} = \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l}\left(\boldsymbol{I} + \frac{\partial \mathcal{F}_l(\boldsymbol{x}_l)}{\partial \boldsymbol{x}_l}\right). \tag{1}$$

Notably, the LayerNorm Jacobian multiplies both the Jacobian along the skip branch $\boldsymbol{I}$ and the derivative along the residual branch $\frac{\partial \mathcal{F}_l(\boldsymbol{x}_l)}{\partial \boldsymbol{x}_l}$. As highlighted by Nguyen and Salazar [15], this leads to instability if $\frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l}$ repeatedly expands or contracts gradients across layers. We can characterise when LayerNorm can act as a contraction (see Appendix C):

$$\left\| \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l} \right\|_2 \leq \frac{\|\boldsymbol{\gamma}\|_\infty}{\sigma(\boldsymbol{z}_l)}. \tag{2}$$

The crucial implication of equation 2, together with equation 1, is that the LayerNorm Jacobian acts as a shared bottleneck for gradient propagation through both the skip and residual branches. In particular, the full sublayer Jacobian is left-multiplied by a factor whose spectral norm is controlled, up to the LayerNorm gain, by the inverse of $\sigma(\boldsymbol{z}_l)$. As post-norm Transformers are scaled up in depth, large hidden-state variance can therefore contract the gradients flowing through the entire sublayer, causing gradient norms to shrink during backpropagation. This finding motivates the design of reliable methods for constraining hidden-state variance during model training.



(a) Variance expansion in vanilla post-norm.

(b) Token alignment in vanilla post-norm.

(c) Token alignment with residual scaling.

(d) Hidden variance with residual scaling.

Figure 1: Effect of residual scaling on training failures observed when training a 12-layer post-norm Transformer according to the setup in Appendix D with learning rate 10^{-2}. We report statistics over the hidden state after the residual addition in the final Attention sublayer. Figures 1a & 1b show the behaviour of vanilla post-norm with residual scaling factor $c = 1$, whereas figures 1c and 1d show the effect of other choices of c.

3. KiteNorm

KiteNorm combines three techniques to improve both stability and performance. Residual scaling, discussed in section 3.1, is essential to our stabilisation strategy and explains the core dynamic of our method. We further introduce variance regularisation, described in section 3.2, which prevents this stability from degrading over time when models are trained with large learning rates. Finally, to improve the expressivity, we leverage scalarised double-normalisation, explained in section 3.3.

3.1. Residual Scaling

Variance expansion and rank collapse. While several prior works [9, 12, 28] have observed the link between gradient vanishing and activation variance in post-norm Transformers, the underlying pressure that causes activation scales to explode at high learning rates remains largely unexplored. We observe that hidden-state variance expansion coincides with rank collapse, shown in figure 1a & 1b. This suggests that the hidden-state scale can keep growing even after token representations become strongly aligned. We comment on the metric in Appendix A.2.

Residual scaling reverses rank collapse. Recent work on rank collapse [10] proposes that weighting the relative contribution of the skip and residual branches in favour of the skip branch can reduce the rate of rank collapse. We observe this in practice by reducing the residual-branch scale, shown in figure 1c & 1d. While the unscaled post-norm model diverges while exhibiting rank collapse, choosing a sufficiently small residual scale can reverse rank collapse before divergence, and this reversal coincides with a reversal of hidden-state expansion.

Equivalence to skip branch scaling. Since the theory literature specifies scaling the skip branch [10], Appendix A.3 derives the equivalence between scaling the skip and residual branches. In practice, we adopt the simple choice $c = \frac{1}{2L}$ following Chen and Wei [3].

3.2. Variance Regularisation

In our experiments, residual scaling is sufficient to induce a stable early-training regime. To prevent later instability, we add a regularisation term based on hidden-state variance to the optimisation loss,

which we call variance regularisation. This additional constraint guards against gradual variance expansion after the early training stage documented in section 3.1. Specifically, we propose the ReLU regulariser term:

$$\text{mean}_l \left(\text{ReLU}(\sigma(\boldsymbol{z}_l)^2 - 1)\right) \tag{3}$$

In practice, the resulting ReLU penalties are averaged across the batch, sequence, and sublayer dimensions before being added to the optimisation loss with weight λ. Notably, this regulariser only affects the loss if the variance of hidden states entering the outer LayerNorm exceeds 1. Hence, the regulariser does not affect optimisation while hidden-state variance remains below this threshold. We use regularisation strength $\lambda = 1$, which is sufficient for stability across our experiments.

3.3. Scalarised Double-Normalisation

Double-norm, following Chen and Wei [3], is an architectural variant of the post-norm topology that we observe to improve model performance in our ablations. Because we hypothesise that the key mechanism behind these gains is the ability to learn separate input scales for the skip and residual branches in each sublayer, we propose using double-norm with scalarised LayerNorm:

$$\text{LN}_{\text{scalarised}}(\boldsymbol{X}) = \gamma' \cdot \tilde{\boldsymbol{X}} + \beta' \tag{4}$$

where $\gamma', \beta' \in \mathbb{R}$ are scalar parameters initialised as 1 and 0, respectively. This restricts LayerNorm to global scale and shift parameters. Empirically, this improves performance.

4. Experiments

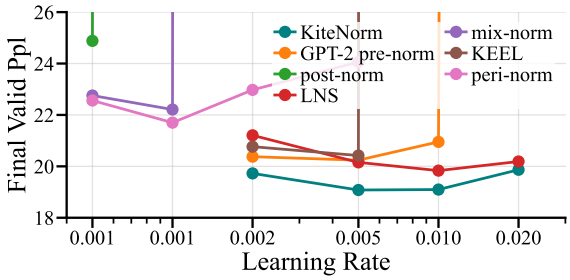
We perform a range of experiments to compare KiteNorm against competitive baselines from existing work. First, in section 4.1, we compare KiteNorm with leading normalisation methods in our setting, evaluating practical performance. Then, in section 4.2, we analyse and compare the scaling properties of KiteNorm against the strongest baseline. Finally, in section 4.3, we document the efficacy of each component of KiteNorm through ablation studies.

We train decoder-only Transformers on SlimPajama-627B [20] using the Chinchilla ratio of 20 training tokens per model parameter [8]. Our optimisation setup uses AdamW [13], fixed $\beta_1 = \beta_2$ following Orvieto and Gower [17], gradient clipping, rotary position embeddings (RoPE) [21], and a GLU feed-forward block [19]. For KiteNorm, we add the ReLU regulariser from equation 3 to the optimisation loss. Appendix D.1 provides the full setup and hyperparameter specification. Unless otherwise specified, all trained models share this setup. For tuned results, we report the run with the lowest final loss or perplexity from the stated learning-rate grid.

4.1. Comparison Against Leading Baselines

Baseline selection. We select competitive baselines for scalable performance in Transformers under our setup: GPT-2 pre-norm [18], LayerNorm-Scaling [23], peri-norm [5, 11], vanilla post-norm [24], KEEL [3], and mix-norm [12]. Appendix D.2 gives the implementation breakdown.

Learning rate robustness. We compare these methods in a 48-layer Transformer, where the challenges of normalisation placement are more pronounced. In figure 2, KiteNorm and LayerNorm-Scaling are the most robust to high learning rates, whereas vanilla post-norm, KEEL, and mix-norm are comparatively sensitive. The odd peri-norm curve is discussed separately in Appendix E.4.



(a) Learning rate exploration.

Method	LR*	Best	Worst
GPT-2 pre-norm	$5 \cdot 10^{-3}$	20.240	20.343
KEEL	$5 \cdot 10^{-3}$	20.316	20.420
Peri-norm	$1 \cdot 10^{-3}$	21.365	21.699
Mix-norm	$1 \cdot 10^{-3}$	22.011	22.211
Post-norm	$5 \cdot 10^{-4}$	24.807	24.882
LayerNorm-Scaling	$1 \cdot 10^{-2}$	19.834	19.970
KiteNorm	$5 \cdot 10^{-3}$	**19.080**	**19.156**

(b) Best and worst values reported from three seeds at the optimal learning rate.

Figure 2: Final validation perplexities of KiteNorm and competitive LayerNorm methods in a 48-layer Transformer.

Seed robustness. We rerun each method twice at its selected best learning rate to obtain a more reliable estimate of the performance difference. KiteNorm maintains a substantial advantage in both the best and worst values across three seeds. Although LayerNorm-Scaling slightly outperforms GPT-2 pre-norm in this 48-layer comparison, Appendix E.5 shows that its performance degrades at 96 layers. We therefore use GPT-2 pre-norm as the more robust baseline below.

4.2. Scalability

Scaling laws. Across depth, training budget, batch size and width, KiteNorm keeps a consistent lead over GPT-2 pre-norm, displayed in figure 4. Appendix A.4 gives the full sweep descriptions. The depth sweep is especially informative because normalisation failures are most pronounced there. Both models improve as depth increases, but KiteNorm remains ahead from shallow to deep settings. The gap also persists when the training budget is varied, where the relative gain is largest at lower budgets, and when the batch size is increased up to 1024, where KiteNorm is more robust. Width changes do not remove the gap.

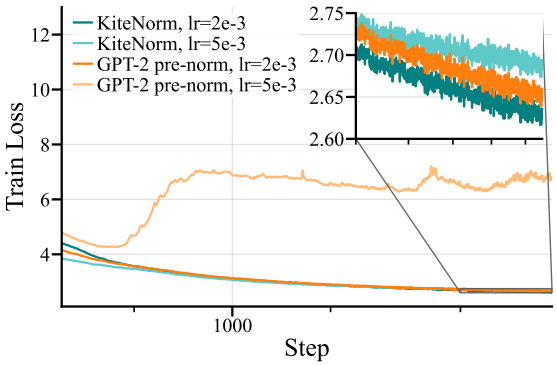
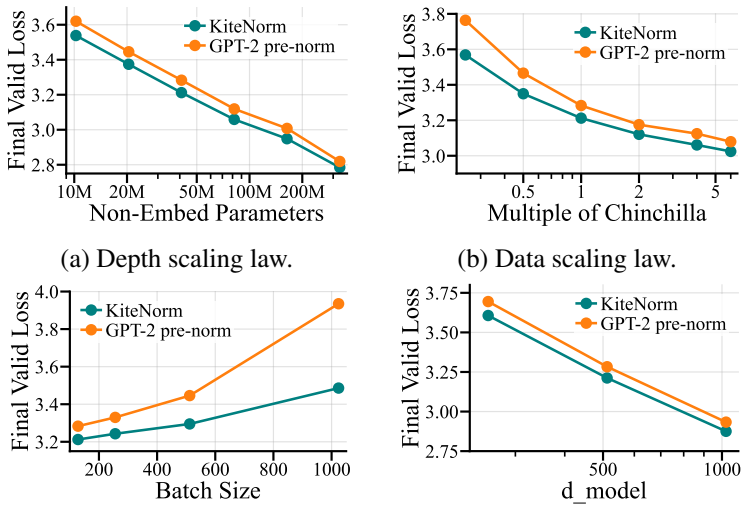


Figure 3: Training loss curves of 1B models in Pythia configuration.

1B test. To test combined scaling, we train a 1B-parameter decoder-only Transformer based on the Pythia 1B configuration [2] with a warmup-stable-decay regime [27], see Appendix D.1 for details. We use 16 layers, $d_{\text{model}} = 2048$, batch size 1024, and otherwise match the sequence length, number of attention heads, and number of GLU parameters to Pythia 1B. Figure 3 shows that KiteNorm achieves lower training loss at learning rate $2 \cdot 10^{-3}$ and remains stable at $5 \cdot 10^{-3}$, where GPT-2 pre-norm becomes unstable. Lastly, Appendix E.6 shows that the overall performance difference is not an artifact of withholding the ReLU regulariser from GPT-2 pre-norm.



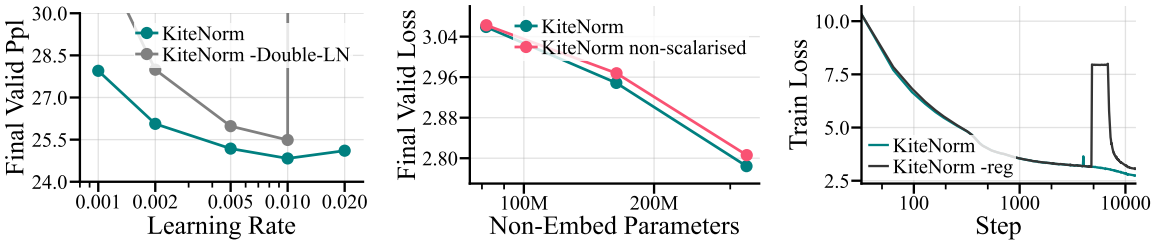
(a) Depth scaling law.

(b) Data scaling law.

(c) Batch size scaling law.

(d) Width scaling law.

Figure 4: Scaling behaviour of KiteNorm and GPT-2 pre-norm across depth, training budget, batch size, and width.



(a) Double-norm impact in 12-layer Transformer.

(b) Scalarised LayerNorm in Transformers at depth.

(c) Variance regularisation in 96-layer Transformer.

Figure 5: Ablation studies on KiteNorm in decoder-only Transformers.

4.3. Ablations

Each component of KiteNorm contributes to the above results, shown in figure 5. Removing the inner LayerNorm reduces KiteNorm to a residually scaled post-norm model and degrades both validation performance and stability. Replacing scalarised LayerNorm with standard LayerNorm is also weaker, with the largest gap appearing in deeper models. Finally, in a 96-layer high-learning-rate stress test, the model diverges without variance regularisation, whereas the regularised model remains stable.

5. Limitations

Our study is limited to decoder-only language models trained on SlimPajama, with the largest combined-scaling test at 1B parameters. We do not test encoder-only, encoder-decoder, multimodal, very long-context, instruction-tuned, or substantially larger settings. The residual scale $c = 1/(2L)$ and variance regularisation weight $\lambda = 1$ are fixed heuristics, and the ReLU penalty is only one possible variance constraint. Our rank-collapse explanation remains partial.

ACKNOWLEDGMENTS

The authors acknowledge the financial support of the Hector Foundation. Antonio Orvieto is supported by the AI2050 program at Schmidt Sciences.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pages 2397–2430. PMLR, 2023.

[3] Chen Chen and Lai Wei. Post-layernorm is back: Stable, expressive, and deep. *arXiv preprint arXiv:2601.19895*, 2026.

[4] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

[5] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835, 2021.

[6] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pages 2793–2803. PMLR, 2021.

[7] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

[8] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

[9] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020.

[10] Federico Arangath Joseph, Jerome Sieber, Melanie N Zeilinger, and Carmen Amo Alonso. Lambda-skip connections: the architectural component that prevents rank collapse. *arXiv preprint arXiv:2410.10609*, 2024.

[11] Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture. *arXiv preprint arXiv:2502.02732*, 2025.

[12] Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*, 2024.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[14] Ido Nachum, Jan Hązła, Michael Gastpar, and Anatoly Khina. A johnson–lindenstrauss framework for randomly initialized cnns. *arXiv preprint arXiv:2111.02155*, 2021.

[15] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th international conference on spoken language translation*, 2019.

[16] 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. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022.

[17] Antonio Orvieto and Robert M Gower. In search of adam's secret sauce. *arXiv preprint arXiv:2505.21829*, 2025.

[18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[19] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[20] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627b token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. Accessed: 2026-05-06.

[21] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[22] Shangwen Sun, Alfredo Canziani, Yann LeCun, and Jiachen Zhu. The spike, the sparse and the sink: Anatomy of massive activations and attention sinks. *arXiv preprint arXiv:2603.05498*, 2026.

[23] Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

[26] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1810–1822, 2019.

[27] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

[28] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

Appendix A. Supplementary Details

A.1. Notation and Rank-Collapse Metric

In this section, we establish notation, provide context for our work, and derive the crucial backward pass dynamics of post-norm Transformers.

Normalised Architectures. Define the normalisation of a vector $\boldsymbol{v} \in \mathbb{R}^d$ as follows: $\tilde{\boldsymbol{v}} = \frac{\boldsymbol{v}-\mu(\boldsymbol{v})\mathbb{1}}{\sigma(\boldsymbol{v})}$, where μ and σ denote the mean and standard deviation, respectively, and $\mathbb{1}$ is the size-d vector of ones. LayerNorm [1], as used in Transformers, can then be written as $\mathrm{LN}(\boldsymbol{X}) = \boldsymbol{\gamma} \cdot \tilde{\boldsymbol{X}} + \boldsymbol{\beta}$, where $\boldsymbol{X} \in \mathbb{R}^{S\times d}$ is a length-S sequence of tokens that are embedded in hidden dimension d. Normalisation is applied row-wise across d. The parameters $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$ act element-wise across d and are called the gain and bias of LayerNorm. Implementations of LayerNorm often add a small constant ϵ in the denominator during the normalisation step to ensure numerical stability, though we omit it for clarity in this section. The original Transformer [24] used normalisation as a component of its sublayers in the following form: $\mathrm{Sublayer}_{\text{post-norm}}(\boldsymbol{X}) = \mathrm{LN}(\boldsymbol{X} + \mathcal{F}(\boldsymbol{X}))$. A decoder-only Transformer stacks such sublayers, with $\mathcal{F}$ alternating between self-attention and an MLP transformation. We denote the number of Transformer layers by L, so that the model has $2L$ sublayers. We write $\boldsymbol{X}_l$ for the hidden states entering the l-th sublayer, where $l \in \{1, 2, \ldots, 2L\}$, and $\mathcal{F}_l$ for the corresponding residual branch transformation. To differentiate sublayer inputs and outputs, we write $\boldsymbol{Z}_l = \boldsymbol{X}_l + \mathcal{F}_l(\boldsymbol{X}_l)$. $\boldsymbol{I}$ denotes the identity matrix.

Matrices and Norms. $\|\cdot\|_q$ describes the L_q norm for vectors, and the corresponding induced operator norm for matrices. $\|\cdot\|_F$ describes the Frobenius norm. The i-th row of a matrix $\boldsymbol{A}_l$ is denoted by $\boldsymbol{A}_{l,i}$. Our theoretical derivations omit the sequence length and batch size. We thus write $\boldsymbol{x}_l \in \mathbb{R}^d$ for a row from $\boldsymbol{X}_l$ and $\boldsymbol{z}_l \in \mathbb{R}^d$ for a row from $\boldsymbol{Z}_l$.

A.2. Rank-Collapse Metric

Noci et al. [16] investigate rank collapse in Transformers by measuring the alignment of token representations using a metric originating from Cho and Saul [4] and Nachum et al. [14]. For two token indices i, i' in the hidden-state matrix $\boldsymbol{X}_l$ output by sublayer l, define the token correlation as

$$\rho_{i,i'}(\boldsymbol{X}_l) = \frac{\mathbb{E}\langle \boldsymbol{X}_{l,i}, \boldsymbol{X}_{l,i'}\rangle}{\sqrt{\mathbb{E}\|\boldsymbol{X}_{l,i}\|_2^2 \mathbb{E}\|\boldsymbol{X}_{l,i'}\|_2^2}}, \tag{5}$$

where the expectation is taken with respect to the model weights at initialisation.

Following Noci et al. [16], we define token alignment as the average token correlation

$$\tau(\boldsymbol{X}_l) = \frac{1}{S(S-1)} \sum_{i\neq i'} \rho_{i,i'}(\boldsymbol{X}_l), \tag{6}$$

where S is the sequence length. Token representations pointing in the same direction have correlation 1. Thus, as token alignment tends to 1, the representation becomes increasingly rank-collapsed. In practice, we compute this metric over the actual model weights, replacing the expectation with the empirical value computed from the trained model states. Although other rank-collapse metrics have been proposed [6], we find that they behave similarly in practice and focus on token alignment because it has a direct geometric interpretation. Token alignment therefore provides a simple empirical measure of rank collapse in Transformers, which we use in the following section to study its connection to gradient vanishing.

A.3. Equivalence of Skip and Residual Scaling

Since the theory literature specifies scaling the skip branch [10], we formalise the equivalence between scaling the skip and residual branches in the following, demonstrating that the residual scaling we propose aligns with prior work.

We express skip scaling a and residual scaling c generally and relate them to each other. For positive a, c and ignoring the stability constant ϵ, it holds that

$$\mathrm{LN}(a\boldsymbol{x}_l + c\mathcal{F}_l(\boldsymbol{x}_l)) = \mathrm{LN}\left(ac\left(\frac{1}{c}\boldsymbol{x}_l + \frac{1}{a}\mathcal{F}_l(\boldsymbol{x}_l)\right)\right) = \mathrm{LN}\left(\frac{1}{c}\boldsymbol{x}_l + \frac{1}{a}\mathcal{F}_l(\boldsymbol{x}_l)\right). \tag{7}$$

Thus, the forward pass depends on the relative scale $\frac{c}{a}$. The effects of the rescaling also appear in the Jacobian. Let $\boldsymbol{z}_l = a\boldsymbol{x}_l + c\mathcal{F}_l(\boldsymbol{x}_l)$. Following equation 1 we have

$$\frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l} = \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l}\left(a\boldsymbol{I} + c\frac{\partial \mathcal{F}_l(\boldsymbol{x}_l)}{\partial \boldsymbol{x}_l}\right). \tag{8}$$

The equivalence of residual and skip scaling becomes apparent here when we examine the upper bound induced by equation 2. Let Var and Cov denote variance and covariance computed across the hidden dimension. For sufficiently large a or small c, we approximate

$$\left\|\frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l}\right\|_2 \leq \frac{\|\boldsymbol{\gamma}\|_\infty}{\sqrt{a^2\mathrm{Var}(\boldsymbol{x}_l) + c^2\mathrm{Var}(\mathcal{F}_l(\boldsymbol{x}_l)) + 2ac\mathrm{Cov}(\boldsymbol{x}_l, \mathcal{F}_l(\boldsymbol{x}_l))}} \approx \frac{\|\boldsymbol{\gamma}\|_\infty}{a\sigma(\boldsymbol{x}_l)}. \tag{9}$$

Substituting this approximation into equation 8, we obtain

$$\frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l} \approx \frac{\|\boldsymbol{\gamma}\|_\infty}{a\sigma(\boldsymbol{x}_l)}\left(a\boldsymbol{I} + c\frac{\partial \mathcal{F}_l(\boldsymbol{x}_l)}{\partial \boldsymbol{x}_l}\right) = \frac{\|\boldsymbol{\gamma}\|_\infty}{\sigma(\boldsymbol{x}_l)}\left(\boldsymbol{I} + \frac{c}{a}\frac{\partial \mathcal{F}_l(\boldsymbol{x}_l)}{\partial \boldsymbol{x}_l}\right). \tag{10}$$

Hence, for positive scales and under the approximation above, increasing skip scaling has an analogous effect to decreasing residual scaling through the ratio $\frac{c}{a}$. This explains why residual scaling achieves the effect of mitigated rank collapse documented in Joseph et al. [10]. In practice, we adopt the simple choice $c = \frac{1}{2L}$ following Chen and Wei [3].

A.4. Scaling-Sweep Details

We analyse the scaling behaviour of KiteNorm against GPT-2 pre-norm across depth, training budget, batch size, and width. Except for the axis being varied, we keep the base configuration fixed. For depth and width sweeps, we adjust the training budget to maintain the Chinchilla-style ratio of 20 tokens per parameter. For each method and sweep point, we report the run with the lowest final validation loss from $(10^{-3}, 2\cdot10^{-3}, 5\cdot10^{-3}, 10^{-2}, 2\cdot10^{-2})$.

Depth scaling. First, we analyse scaling across depth, where normalisation methods often exhibit their most pronounced failures. To that end, we start with a depth of 3 layers and double it successively until reaching 96 layers. Measuring the final validation loss of each point, we obtain the scaling law displayed in figure 4a.

Comparing KiteNorm and GPT-2 pre-norm in terms of plotting the best final validation loss against the number of non-embedding parameters, we observe that both continue to gain in performance significantly as depth increases. However, KiteNorm maintains a lead over GPT-2 pre-norm

both in shallow and deep models. A narrowing gap in validation loss should not be interpreted as diminishing practical relevance. At lower loss values, comparable absolute improvements can require substantially larger increases in training compute.

Extending the scaling law analysis to data, batch size, and model width, we obtain the additional scaling laws shown in figure 4.

Training budget scaling. Inspecting the impact of training budget on final model performance, we vary the training budget as multiples of the Chinchilla rule of 20 tokens per parameter. Specifically, we test the multipliers ($\frac{1}{4}$, $\frac{1}{2}$, 1, 2, 4, 6), obtaining the data scaling law in figure 4b. Here, too, KiteNorm demonstrates superior performance across lower and higher budgets, with especially large relative gains under low budgets.

Batch size scaling. Since larger batch sizes are commonly used to improve hardware utilisation in decoder-only Transformer training, we investigate the performance impact of doubling our batch size from the base value of 128 up to 1024. The resulting batch size scaling law is shown in figure 4c. KiteNorm demonstrates greater robustness to large batch sizes.

Width scaling. Finally, for completeness, we investigate the scaling of final performance across width by both doubling and halving the hidden representation size d_{model}, displaying the resulting width scaling law in figure 4d. The performance gap favouring KiteNorm is robust to these changes in width.

Combined scaling. To test whether these trends persist when depth, width, batch size, and training budget are scaled together, we train a 1B-parameter decoder-only Transformer based on the Pythia 1B configuration [2]. Specifically, we use 16 layers, $d_{\text{model}} = 2048$, and batch size 1024, with sequence length, number of Attention heads, and number of GLU parameters matched to the Pythia 1B specification. For these runs, we use a warmup-stable-decay learning rate schedule [27] with no decay and train with the Chinchilla-rule budget.

The resulting training loss curves are shown in figure 3. KiteNorm achieves lower training loss at learning rate $2 \cdot 10^{-3}$ and remains stable at learning rate $5 \cdot 10^{-3}$, where GPT-2 pre-norm becomes unstable.

Finally, we note that the comparison is not an artefact of omitting variance regularisation from GPT-2 pre-norm, as Appendix E.6 shows that the ReLU regulariser does not improve GPT-2 pre-norm.

Appendix B. Related Works

Normalisation placement has played a central role in the development of Transformer architectures. The original Transformer used post-norm sublayers, but many large-scale implementations have since adopted pre-norm following Wang et al. [26]. In pre-norm Transformers, LayerNorm is placed before the residual branch transformation:

$$\text{Sublayer}_{\text{pre-norm}}(\boldsymbol{X}) = \boldsymbol{X} + \mathcal{F}(\text{LN}(\boldsymbol{X})). \tag{11}$$

Unlike post-norm, pre-norm leaves the skip connection uninterrupted during backpropagation, allowing gradients to flow directly across depth [26]. Subsequent work linked this structure to improved gradient conditioning in expectation [28].

Pre-norm scaling methods. The stability of pre-norm has motivated several methods for controlling signal growth along the skip branch. GPT-2 applies residual output scaling by initialising the output weights of each sublayer with an additional factor of $1/\sqrt{2L}$ [18], which compensates for the accumulation of residual updates in deeper models. Later work formalised the importance of such scaling mechanisms, showing that unconstrained signal growth in pre-norm Transformers can reduce the effectiveness of deep layers [23]. Instead of scaling output weights, LayerNorm-Scaling rescales the input to each sublayer after pre-normalisation by a depth-dependent factor.

Additional normalisation. Related approaches such as peri-norm [5, 11] add a second LayerNorm after the residual branch transformation, yielding peri-norm sublayers of the form

$$\text{Sublayer}_{\text{peri-norm}}(\boldsymbol{X}) = \boldsymbol{X} + \text{LN}(\mathcal{F}(\text{LN}(\boldsymbol{X}))). \tag{12}$$

This constrains the magnitude of each residual update before it is added to the skip branch.

Despite these improvements, pre-norm retains architectural limitations. Because each sublayer adds directly to the skip branch, signal accumulation cannot be fully avoided, even when it is mitigated by scaling or additional normalisation. Recent work has also shown that large activations can propagate through pre-norm models at scale [22]. These issues have motivated renewed interest in post-norm architectures, where LayerNorm is applied after the residual addition and therefore normalises both the skip and residual branches.

Post-norm stabilisation. Several recent methods attempt to recover the stability of post-norm while preserving its performance advantages. Mix-norm constructs models from an initial block of post-norm layers followed by pre-norm layers [12]. This reduces the number of post-norm layers over which gradients can vanish, while also reducing the number of pre-norm layers over which signal can accumulate. However, the resulting architecture remains a compromise between the two regimes. DeepNet instead retains a post-norm structure and introduces residual and skip branch scaling to improve optimisation conditions at initialisation [25]. Although effective for very deep encoder-decoder Transformers, we find that such initialisation-based stabilisation is less robust in decoder-only settings trained with large learning rates.

Most recently, Chen and Wei [3] proposed KEEL, a post-norm method that improves decoder-only Transformer training through residual scaling and double normalisation:

$$\text{Sublayer}_{\text{double-norm}}(\boldsymbol{X}) = \text{LN}(\boldsymbol{X} + \mathcal{F}(\text{LN}(\boldsymbol{X}))). \tag{13}$$

Although the additional normalisation may appear redundant, it allows the model to separately learn input scales for the skip and residual branches within each sublayer. Overall, while KEEL improves the stability of post-norm training, we find that it still underperforms strong pre-norm baselines in the decoder-only setting, leaving room for more robust post-norm normalisation methods.

Appendix C. Proof of Eq. 2

We begin with the elementary components of the LayerNorm Jacobian. For brevity, we write μ_l, σ_l to mean $\mu(\boldsymbol{z}_l), \sigma(\boldsymbol{z}_l)$. We ignore the small stability constant ϵ added in implementation.

First, consider the Jacobian of the mean subtraction:

$$\frac{\partial(\boldsymbol{z}_l - \mu_l \mathbb{1})}{\partial \boldsymbol{z}_l} = \boldsymbol{I} - \mathbb{1}\frac{\partial \mu_l}{\partial \boldsymbol{z}_l} = \boldsymbol{I} - \mathbb{1}\frac{1}{d}\mathbb{1}^\top = \boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top. \tag{14}$$

We also require the Jacobian of the standard deviation:

$$
\begin{aligned}
\frac{\partial \sigma_l}{\partial \boldsymbol{z}_l} &= \frac{\partial}{\partial \boldsymbol{z}_l}\left(\sqrt{\frac{1}{d}\left(\boldsymbol{z}_l - \mu_l \mathbb{1}\right)^\top \left(\boldsymbol{z}_l - \mu_l \mathbb{1}\right)}\right) \\
&= \frac{1}{2\sigma_l}\frac{\partial}{\partial \boldsymbol{z}_l}\left(\frac{1}{d}\left(\boldsymbol{z}_l - \mu_l \mathbb{1}\right)^\top \left(\boldsymbol{z}_l - \mu_l \mathbb{1}\right)\right) \\
&= \frac{1}{2\sigma_l}\cdot\frac{2}{d}(\boldsymbol{z}_l - \mu_l \mathbb{1})^\top \frac{\partial}{\partial \boldsymbol{z}_l}(\boldsymbol{z}_l - \mu_l \mathbb{1}) \\
&= \frac{1}{2\sigma_l}\cdot\frac{2}{d}\left(\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top\right)(\boldsymbol{z}_l - \mu_l \mathbb{1})\right)^\top \\
&= \frac{1}{d\sigma_l}\left((\boldsymbol{z}_l - \mu_l \mathbb{1}) - \frac{1}{d}\mathbb{1}\mathbb{1}^\top(\boldsymbol{z}_l - \mu_l \mathbb{1})\right)^\top \\
&= \frac{1}{d\sigma_l}\left((\boldsymbol{z}_l - \mu_l \mathbb{1}) - \frac{1}{d}\mathbb{1}(\mathbb{1}^\top \boldsymbol{z}_l - \mathbb{1}^\top(\frac{1}{d}\mathbb{1}^\top \boldsymbol{z}_l)\mathbb{1})\right)^\top \\
&= \frac{1}{d\sigma_l}\left((\boldsymbol{z}_l - \mu_l \mathbb{1}) - \frac{1}{d}\mathbb{1}(\mathbb{1}^\top \boldsymbol{z}_l - (\frac{1}{d}\mathbb{1}^\top \boldsymbol{z}_l d))\right)^\top \\
&= \frac{1}{d\sigma_l}(\boldsymbol{z}_l - \mu_l \mathbb{1})^\top. \qquad (15)
\end{aligned}
$$

Let again $\tilde{\boldsymbol{z}}_l = \frac{\boldsymbol{z}_l - \mu_l \mathbb{1}}{\sigma_l}$. We can now calculate the Jacobian of normalisation:

$$
\begin{aligned}
\frac{\partial \tilde{\boldsymbol{z}}_l}{\partial \boldsymbol{z}_l} &= \frac{\partial}{\partial \boldsymbol{z}_l}\left(\sigma_l^{-1}(\boldsymbol{z}_l - \mu_l \mathbb{1})\right) \\
&= \sigma_l^{-1}\frac{\partial(\boldsymbol{z}_l - \mu_l \mathbb{1})}{\partial \boldsymbol{z}_l} + (\boldsymbol{z}_l - \mu_l \mathbb{1})\frac{\partial(\sigma_l^{-1})}{\partial \boldsymbol{z}_l} \\
&= \sigma_l^{-1}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top\right) + (\boldsymbol{z}_l - \mu_l \mathbb{1})\left(-\sigma_l^{-2}\frac{\partial \sigma_l}{\partial \boldsymbol{z}_l}\right) \\
&= \sigma_l^{-1}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top\right) - (\boldsymbol{z}_l - \mu_l \mathbb{1})\sigma_l^{-2}\left(\frac{1}{d\sigma_l}(\boldsymbol{z}_l - \mu_l \mathbb{1})^\top\right) \\
&= \frac{1}{\sigma_l}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top\right) - \frac{1}{d\sigma_l^3}(\boldsymbol{z}_l - \mu_l \mathbb{1})(\boldsymbol{z}_l - \mu_l \mathbb{1})^\top \\
&= \frac{1}{\sigma_l}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d\sigma_l^2}(\boldsymbol{z}_l - \mu_l \mathbb{1})(\boldsymbol{z}_l - \mu_l \mathbb{1})^\top\right) \\
&= \frac{1}{\sigma_l}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d\sigma_l^2}(\tilde{\boldsymbol{z}}_l \sigma_l)(\tilde{\boldsymbol{z}}_l \sigma_l)^\top\right) \\
&= \frac{1}{\sigma_l}\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l \tilde{\boldsymbol{z}}_l^\top\right). \qquad (16)
\end{aligned}
$$

Finally, consider the Jacobian of LayerNorm itself:

$$
\frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l} = \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \tilde{\boldsymbol{z}}_l}\frac{\partial \tilde{\boldsymbol{z}}_l}{\partial \boldsymbol{z}_l} = \mathrm{Diag}(\boldsymbol{\gamma})\frac{\partial \tilde{\boldsymbol{z}}_l}{\partial \boldsymbol{z}_l} = \frac{1}{\sigma_l}\mathrm{Diag}(\boldsymbol{\gamma})\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l \tilde{\boldsymbol{z}}_l^\top\right) \qquad (17)
$$

Since we are interested in how this Jacobian may contract the gradient, we analyse its spectral norm:

$$\left\| \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l} \right\|_2 \leq \frac{1}{\sigma_l} \cdot \|\mathrm{Diag}(\boldsymbol{\gamma})\|_2 \cdot \left\| \boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top \right\|_2 \tag{18}$$

It is trivial that $\|\mathrm{Diag}(\boldsymbol{\gamma})\|_2 = \|\boldsymbol{\gamma}\|_\infty$. Regarding $\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top$, we prove in the following that this is an orthogonal projection matrix.

Since $\tilde{\boldsymbol{z}}_l$ is normalised, we have $\mathbb{1}^\top\tilde{\boldsymbol{z}}_l = 0$ and $\|\tilde{\boldsymbol{z}}_l\|_2^2 = d$. Likewise, $\|\mathbb{1}\|_2^2 = d$. First we show that this implies $\frac{1}{d}\mathbb{1}\mathbb{1}^\top$ is an orthogonal projector:

$$\left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right)^2 = \frac{1}{d^2}\mathbb{1}\mathbb{1}^\top\mathbb{1}\mathbb{1}^\top = \frac{1}{d^2}\mathbb{1}(d)\mathbb{1}^\top = \frac{1}{d}\mathbb{1}\mathbb{1}^\top \tag{19}$$

As symmetry is trivial, $\frac{1}{d}\mathbb{1}\mathbb{1}^\top$ is an orthogonal projector. It is also trivial that the same argument holds for $\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top$. Thus they are both orthogonal projectors.

We additionally show that they are also orthogonal to each other, using $\mathbb{1}^\top\tilde{\boldsymbol{z}}_l = 0$:

$$\left(\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right)\left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right) = \frac{1}{d^2}\tilde{\boldsymbol{z}}_l(0)\mathbb{1}^\top = 0 = \frac{1}{d^2}\mathbb{1}(0)\tilde{\boldsymbol{z}}_l^\top = \left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right)\left(\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right) \tag{20}$$

Finally, we use this to show that $\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top$ itself is idempotent:

$$\begin{aligned}
\left(\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right)^2 &= \boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top \\
&- \frac{1}{d}\mathbb{1}\mathbb{1}^\top + \left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right)^2 + \left(\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right)\left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right) \\
&- \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top + \left(\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right)^2 + \left(\frac{1}{d}\mathbb{1}\mathbb{1}^\top\right)\left(\frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top\right) \\
&= \boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top \\
&- \frac{1}{d}\mathbb{1}\mathbb{1}^\top + \frac{1}{d}\mathbb{1}\mathbb{1}^\top + 0 \\
&- \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top + \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top + 0 = \boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top
\end{aligned} \tag{21}$$

Hence, considering that any sum of symmetric matrices is itself also symmetric, $\boldsymbol{I} - \frac{1}{d}\mathbb{1}\mathbb{1}^\top - \frac{1}{d}\tilde{\boldsymbol{z}}_l\tilde{\boldsymbol{z}}_l^\top$ is an orthogonal projection matrix. Its eigenvalues are therefore contained in $\{0, 1\}$, so its spectral norm is at most 1.

Consequently, we obtain an upper bound on the spectral norm of the LayerNorm Jacobian:

$$\left\| \frac{\partial \mathrm{LN}_l(\boldsymbol{z}_l)}{\partial \boldsymbol{z}_l} \right\|_2 \leq \frac{\|\boldsymbol{\gamma}\|_\infty}{\sigma(\boldsymbol{z}_l)} \tag{22}$$

HYPERPARAMETER	SHARED VALUE
training corpus	SlimPajama-627B
training objective	next-token cross entropy
sequence length	2048
vocabulary size	50280
d_{model}	512
d_{GLU}	1536
number of heads	8
sublayers per Transformer layer	Attention followed by GLU
Attention	causal self-Attention with RoPE
RoPE base	500000
GLU nonlinearity	SiLU gate
linear projections	no bias
tied token embeddings	False
dropout	none
batch size	128 sequences per optimiser step
optimiser	AdamW
scheduler	linear warm-up followed by cosine decay
warm-up fraction	6%
warm-up start lr	$1 \cdot 10^{-10}$
decay end lr	0.0
weight decay	0.1 on matrix parameters and 0.0 on bias and normalisation parameters
dtype	bfloat16
TF32	enabled
β_1	0.95
β_2	0.95
LayerNorm ϵ	$1 \cdot 10^{-6}$
gradient clipping threshold	1.0
random seed	7 unless reseeding is specified
sampler seed	7 unless reseeding is specified

Table 1: List of shared hyperparameters.

Appendix D. Specification

D.1. Training Setup Details

All models except the 1B models shown in figure 3 were trained on up to 8 NVIDIA A100-SXM4-80GB GPUs. The 1B models were trained on 4 NVIDIA B200 GPUs. Altogether, approximately 1000 NVIDIA A100-SXM4-80GB GPU hours and 50 NVIDIA B200 GPU hours were used for our experiments.

In table 1 we state the baseline hyperparameters shared between all configurations unless otherwise stated. The learning rate is not a shared hyperparameter, because it is the parameter tuned in the learning-rate explorations.

The batch size in table 1 denotes the effective batch size after distributed training and gradient accumulation. Thus, for the 48-layer comparison against leading baselines we use four GPUs, micro-batch size 16 per GPU, and two gradient accumulation steps. For the 12-layer base configuration

we use one GPU, micro-batch size 32, and four gradient accumulation steps. The default 12-layer and 48-layer configurations use step budgets 6866 and 12207 respectively, corresponding to the Chinchilla-style token budgets used throughout the paper. The 1B configurations override the shared architectural and batch-size values with $L = 16$, $d_{\text{model}} = 2048$, batch size 1024, Chinchilla-rule step budget 9537. They use the same sequence length, vocabulary size, number of heads, GLU nonlinearity, optimiser, dtype, weight decay, and LayerNorm ϵ as the smaller models. The GLU width is chosen to match the parameter count of the MLP in the Pythia 1B specification. The 1B models use the warm-up-stable-decay schedule described in Appendix A.4.

D.2. Implementation Details

We distinguish the Transformer layer index $k \in \{1, \dots, L\}$ from the sublayer index $l \in \{1, \dots, 2L\}$. Each Transformer layer contains one Attention sublayer and one GLU sublayer. We write the residual branch function of sublayer l as $\mathcal{F}_l$, and define

$$\mathrm{Mix}_{a,c}(\boldsymbol{U}, \boldsymbol{V}) = a\boldsymbol{U} + c\boldsymbol{V}. \tag{23}$$

Fixed branch scalings are applied in the forward pass through $\mathrm{Mix}_{a,c}$, whereas initialisation scalings are applied once to the output projection matrices immediately after weight initialisation. Unless otherwise stated, all linear and embedding weights are initialised from $\mathcal{N}(0, 0.02^2)$, all LayerNorm scale parameters are initialised to 1, and all LayerNorm shift parameters are initialised to 0.

LayerNorm statistics are computed over the hidden dimension without Bessel correction. Standard LayerNorm uses vector-valued parameters $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{d_{\text{model}}}$ when the shift is enabled. Scalarised LayerNorm uses scalar parameters $\gamma', \beta' \in \mathbb{R}$ as in equation 4. Matching equation 3, the implemented ReLU variance regulariser used by KiteNorm is

$$\mathcal{R}_{\text{ReLU}} = \frac{1}{2L} \sum_{l=1}^{2L} \mathbb{E}_{b,t} \left[\mathrm{ReLU}\left(\sigma(\boldsymbol{z}_{l,b,t})^2 - 1 \right) \right], \tag{24}$$

where $\boldsymbol{z}_l$ is the hidden state immediately after the residual addition and immediately before any outer normalisation in sublayer l, and $\sigma(\cdot)^2$ is the variance over the hidden dimension. The training loss for KiteNorm is $\mathcal{L}_{\text{CE}} + \lambda \mathcal{R}_{\text{ReLU}}$ with $\lambda = 1$. No other baseline in the main comparison against leading baselines uses variance regularisation.

Table 2 states the exact implementation of the scaling and normalisation schemes used in the comparison of competitive approaches.

The learning-rate exploration in figure 2 varies only the maximum learning rate of the scheduler. All other hyperparameters are held fixed according to tables 1 and 2. The reseeded runs use the best learning rate from the displayed grid for each method.

METHOD	SUBLAYER IMPLEMENTATION	SCALING DETAILS
GPT-2 pre-norm	$\boldsymbol{X}_{l+1} = \boldsymbol{X}_l + \mathcal{F}_l(\mathrm{LN}(\boldsymbol{X}_l))$, with a final output LayerNorm. Standard LayerNorm includes the shift parameter.	Runtime skip and residual scales are both 1. The Attention output projection and GLU output projection in every Transformer layer are multiplied at initialisation by $\frac{1}{\sqrt{2L}}$.
LayerNorm-Scaling	Pre-norm with LN replaced in both sublayers of Transformer layer k by $\mathrm{LNS}_k(\boldsymbol{X}) = k^{-1/2}\mathrm{LN}(\boldsymbol{X})$. The final output normalisation is standard LayerNorm, not LayerNorm-Scaling.	Runtime skip and residual scales are both 1. No residual output projection scaling is applied at initialisation. Standard LayerNorm includes the shift parameter.
Post-norm	$\boldsymbol{X}_{l+1} = \mathrm{LN}(\boldsymbol{X}_l + \mathcal{F}_l(\boldsymbol{X}_l))$, without a final output LayerNorm. Standard LayerNorm includes the shift parameter.	Runtime skip and residual scales are both 1. No residual output projection scaling is applied at initialisation.
KEEL	Double-norm: $\boldsymbol{X}_{l+1} = \mathrm{LN}(\mathrm{Mix}_{a_l,1}(\boldsymbol{X}_l, \mathcal{F}_l(\mathrm{LN}(\boldsymbol{X}_l))))$. The outer LayerNorm after the first Attention sublayer is omitted. Standard LayerNorm is used without the shift parameter.	For the two sublayers belonging to Transformer layer $k = 1$, $a_l = 1$. For all sublayers belonging to Transformer layers $k \geq 2$, $a_l = 2L$. The residual branch scale is always 1. No residual output projection scaling is applied at initialisation. The main comparison uses GLUs. The equal-parameter MLP variant is only used in Appendix E.2.
Peri-norm	$\boldsymbol{X}_{l+1} = \boldsymbol{X}_l + \mathrm{LN}(\mathcal{F}_l(\mathrm{LN}(\boldsymbol{X}_l)))$. There is no final output LayerNorm. Standard LayerNorm includes the shift parameter.	Runtime skip and residual scales are both 1. No residual output projection scaling is applied at initialisation.
Mix-norm	The first $\lfloor 0.25L \rfloor$ Transformer layers are post-norm layers and the remaining $L - \lfloor 0.25L \rfloor$ Transformer layers are pre-norm layers. A final output LayerNorm is used because the model ends with a pre-norm suffix. Standard LayerNorm includes the shift parameter.	Runtime skip and residual scales are both 1. Let $L_{\text{pre}} = L - \lfloor 0.25L \rfloor$. Only the Attention output projection and GLU output projection in the pre-norm suffix are multiplied at initialisation by $\frac{1}{\sqrt{2L_{\text{pre}}}}$, following the partial GPT-2 initialisation selected in Appendix E.3.
KiteNorm	Scalarised double-norm: $\boldsymbol{X}_{l+1} = \mathrm{LN}_{\text{scalarised}}\left(\boldsymbol{X}_l + \frac{1}{2L}\mathcal{F}_l(\mathrm{LN}_{\text{scalarised}}(\boldsymbol{X}_l))\right)$. There is no final output LayerNorm.	The skip branch scale is 1 and the residual branch scale is $\frac{1}{2L}$ in every sublayer. No residual output projection scaling is applied at initialisation. The ReLU variance regulariser in equation 24 is used with $\lambda = 1$.

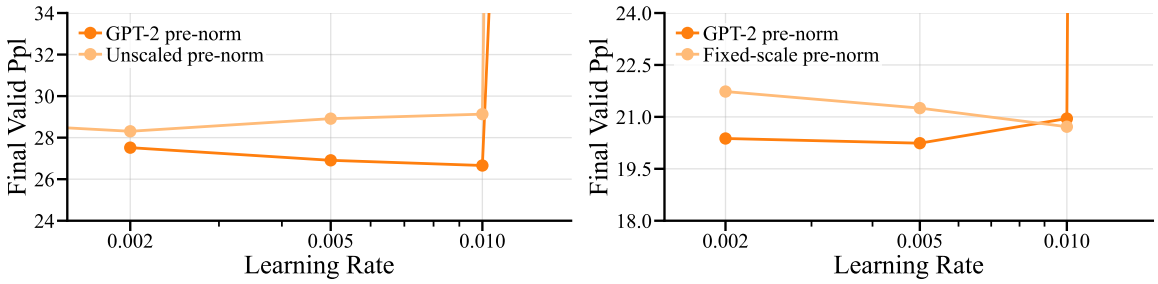
Table 2: Exact implementation of the normalisation and scaling schemes in the comparison of competitive approaches.

Appendix E. Supplementary Experiments

Unless otherwise specified, all models use the setup and hyperparameters given in Appendix D.

E.1. Residual Scaling Variants of Pre-Norm

In order to identify the strongest pre-norm baseline to compare against, we conduct learning rate exploration with unscaled pre-norm and a reparameterisation of GPT-2 pre-norm that applies the scaling in the sublayer, rather than at initialisation. Comparing the result against GPT-2 pre-norm, we obtain figures 6a and 6b respectively. For unscaled pre-norm, we note reduced robustness to high learning rates and decreased performance. For pre-norm with fixed scaling, we observe deteriorated performance at depth. These findings motivate us to compare against GPT-2 pre-norm.



(a) Comparison against unscaled pre-norm in a 12-layer Transformer.

(b) Comparison against fixed scale pre-norm in a 48-layer Transformer.

Figure 6: Learning-rate explorations for pre-norm variants.

E.2. GLU Variant of KEEL

We compare the performance of a KEEL model with our GLU to one with an MLP, where both have the same number of parameters. The result is presented in figure 7, where we find that the GLU variant performs better.

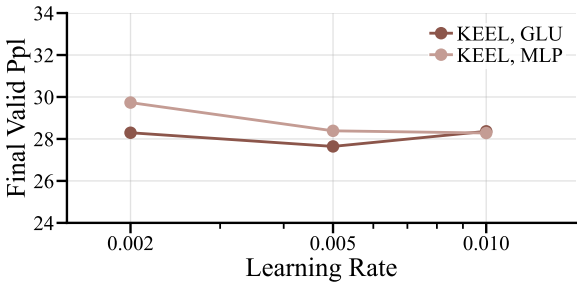


Figure 7: Comparison between KEEL models with GLU and models with MLP in a 12-layer Transformer.

E.3. Residual Scaling for Mix-Norm

Given that GPT-2 style residual scaling can improve pre-norm methods, we evaluate whether this also applies to the pre-norm section of mix-norm models. Here, we compare vanilla mix-norm against a variant where GPT-2 style residual scaling is applied in the pre-norm sublayers, displaying the result in figure 8. Since we observe a clear performance improvement when using GPT-2 style residual scaling, we apply it in all other mix-norm runs.

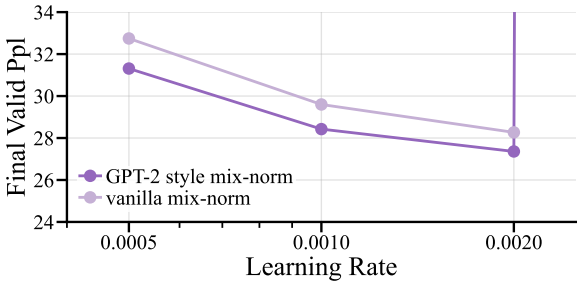


Figure 8: Comparison between vanilla mix-norm and mix-norm with GPT-2 style residual scaling in a 12-layer Transformer.

E.4. Rank Collapse with Peri-Norm

Since we find the learning rate exploration yields an oddly non-hyperbolic curve for peri-norm, we analyse its dynamics and find that peri-norm is prone to rank collapse. We show this empirically by analysing the token alignment in a peri-norm model trained with a learning rate of 10^{-2}, seen in figure 9. Indeed, while the first layer is often an exception in Transformer models, the subsequent layers show a clear tendency to rank collapse, explaining the poor performance of peri-norm in our experiments.

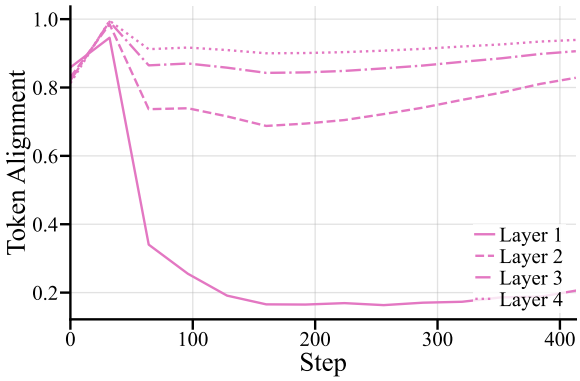


Figure 9: Rank collapse in hidden states after the residual addition in Attention sublayers when training a 12-layer Transformer with peri-norm.

E.5. LayerNorm-Scaling at Very High Depth

While our comparison of 48-layer Transformers in section 4.1 suggests that LayerNorm-Scaling outperforms GPT-2 pre-norm, we suspect that this gap does not hold across depth. We therefore compare LayerNorm-Scaling against GPT-2 pre-norm in an equivalent Transformer with double the number of layers and show the result of training in figure 10. We observe that the performance gap favouring LayerNorm-Scaling vanishes at this level of depth.

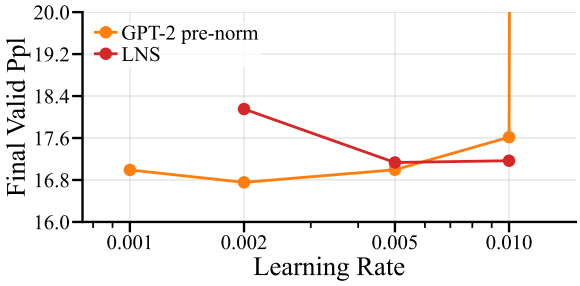


Figure 10: Comparison between GPT-2 pre-norm and LayerNorm-Scaling in a 96-layer Transformer.

E.6. GPT-2 Pre-Norm With the ReLU Regulariser

We explore whether our ReLU regulariser has the potential to improve the GPT-2 pre-norm baseline to ensure a fair comparison. In a 12-layer Transformer, we train GPT-2 pre-norm with the ReLU regulariser enabled, yielding the loss curves displayed in figure 11. The result shows clearly elevated instability when enabling variance regularisation while training GPT-2 pre-norm.

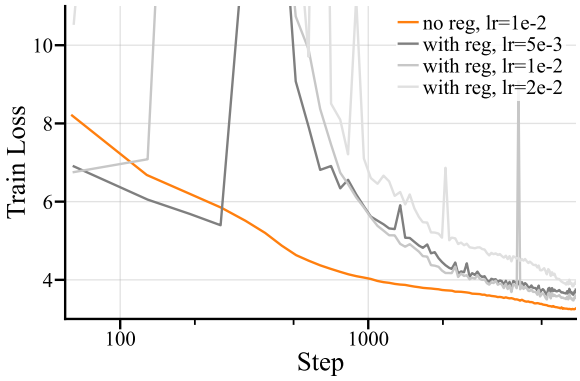


Figure 11: Comparison between training of GPT-2 pre-norm with and without the ReLU regulariser enabled in a 12-layer Transformer.