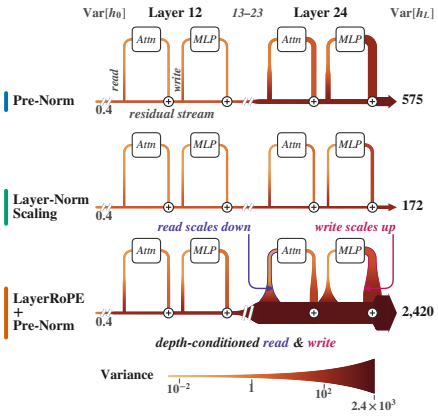
LAYERROPE: DYNAMIC DEPTH-WISE MAGNITUDE & ANGULAR SUPERPOSITION

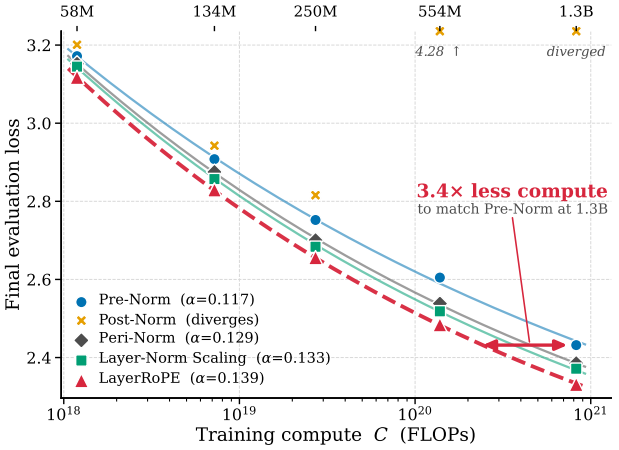
Anonymous authors
Paper under double-blind review

ABSTRACT

As data propagates through a Transformer, the norm of its hidden states grows by orders of magnitude with depth, a phenomenon framed as 'curse of depth' and nearly universally treated as a pathology to be suppressed. We take the opposite view. Across 16 pre-trained LLMs from 9 families, spanning dense, mixture-of-experts and hybrid architectures and Pre-, Peri- and Post-Norm designs, we find that this growth reflects an emergent *depth-positional encoding*, carried by the only learned per-layer gain on the residual stream, the normalization weight γ: with depth, γ grows in magnitude and rotates in direction, jointly encoding the layer index. We make this depth-conditioned encoding explicit with **LayerRoPE**, an implicit analog of RoPE along the depth axis, which replaces all layerwise γ vectors with a single shared vector, depth-conditioned scalars, at a net reduction in parameters and $<0.02\%$ change in FLOPs. Across a model ladder scaled upto 100B+ tokens, LayerRoPE consistently outperforms Pre-, Post- and Peri-Norm, Layer-Norm Scaling and DeepNet, reaching Pre-Norm's 1.3B loss with $3.4\times$ less compute; LayerRoPE is the only approach whose loss improves monotonically with strong convergence, as depth scales to 512 layers. It improves learning-rate sensitivity by 3–$10\times$, and transfers naively to and consistently improves looped latent models and Vision Transformers. Inspecting its learned schedule inverts the prevailing premise: LayerRoPE does not shrink the residual stream but *widens* it, damping what each block reads while amplifying what it writes. Depth stability, our results suggest, calls not for suppressing the residual stream, but for depth-conditioned regulation of the computational blocks it feeds.



(a) Illustration: LayerRoPE's regulation of computational block with *depth-conditioned* I/O, while *maximizing residual variance*.



(b) Compute-scaling curves on a $4\times$ Chinchilla model-ladder

Figure 1: **LayerRoPE offers a contrary premise:** *Maximizing residual variance with depth-conditioned regulation of computational blocks*, allows better convergence, deeper networks, wider learning basins, efficiently ($<0.02\%$ FLOP increase), and across architectures and modalities.

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.

1 INTRODUCTION

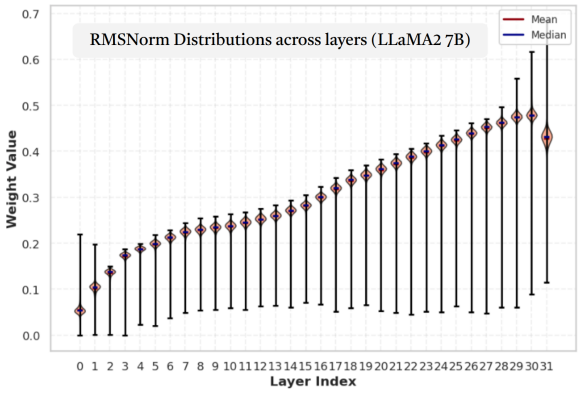
Transformers underpin modern AI, from large language models (LLMs) to Vision Transformers (ViTs). All such systems share a persistent pathology: as the input propagates through the network, the ℓ_2 norms of hidden states grow rapidly with depth, often by several orders of magnitude between the input embeddings and the final layers. Recent studies have characterized various distinct manifestations of this growth. Sun et al. (2024) identify "massive activations" whose magnitudes exceed the mean by more than 3000 times; Darcet et al. (2024) observe high-norm "register" tokens that emerge in ViTs to absorb global context; and Sun et al. (2025) frame the broader phenomenon of activation variance growth in Pre-Norm transformers as a "curse of depth" that degrades gradient flow and layer contributions in deeper LLMs.

The prevailing beliefs treat this growth as something to suppress or re-engineer away, along two lines. **One**, damps the residual stream directly: Layer-Norm Scaling (Sun et al., 2025), for instance, scales each block's input, the output of its layer normalization, by $1/\sqrt{\ell+1}$, and Peri-Norm (Kim et al., 2025) re-normalizes each sub-layer's output before it re-enters the stream. **Second**, to rewrite the residual pathway itself: Attention Residuals (Team et al., 2026) show that the unit-weight residual is equivalent to a low-rank linear attention over depth, whose uniform accumulation forces hidden-state norms to grow as $O(L)$, and replace it with a learned aggregation that holds magnitudes bounded; Hyper-Connections (Zhu et al., 2025) and Manifold-Constrained Hyper-Connections (Xie et al., 2025) instead widen the single stream into several parallel streams, constraining their mixing to preserve signal magnitude. A detailed discussion of related work is provided in Section F. Nonetheless, across both lines, the unifying premise is that activation growth is a defect to be removed.

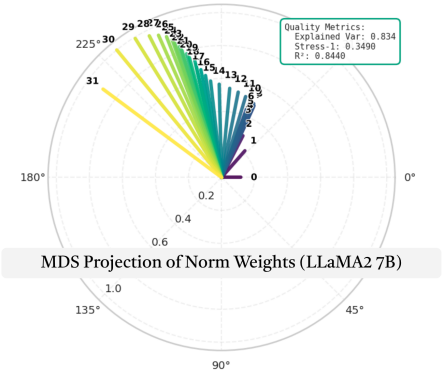
We take the opposite view: The residual growth is not in itself a failure mode, and the goal must not be to shrink the residual pipe that flows through the network. Rather, to regulate the computational blocks it feeds into, allowing us to further widen the residual pipe flowing through the network. We show through broad experimentation that this view leads to better depth scaling, than residual damping or unconstrained computation. In order to regulate computational blocks to read/write into a wide residual stream, it is first necessary to ask what the growth *encodes*. We find, interestingly, that the systemic, per-layer growth of activation norms is indicative of an emergent *depth-positional encoding* that the network learns to express through the only learned per-layer gain directly on the residual stream: the RMSNorm weight γ. Through analysis of a large breadth of LLMs, we show that the MLP block's γ vectors grow and rotate conditioned variously on the network depth. Magnitude and rotation, we hypothesize, therefore, are conditioned on the residual stream's depth, and therefore jointly encode a form of *layer-index encoding*: an implicit analog of RoPE (Su et al., 2024) along the depth axis rather than along sequence position.

We make this layer-index encoding explicit. **LayerRoPE** replaces the per-layer gains γ_ℓ with one shared γ per site, rotated and scaled by a few depth-conditioned scalars. It is a drop-in addition to any computational block, at net fewer parameters and a $<0.02\%$ change in FLOPs. Our contributions are:

- **An emergent depth code.** Across 16 LLMs with varying families and architectures, the norm weights γ show depth-conditioned magnitude and rotation, with $\sim$99% of its dimensions following a shared layer-wise trajectory (Section 2).
- **Better scaling.** On a 58M–1.3B ladder trained at $4\times$ the Chinchilla budget (up to 107B tokens), LayerRoPE attains the lowest loss at every scale, reaches Pre-Norm's 1.3B loss with $3.4\times$ less compute, and achieves the best average zero-shot accuracy (Section 4.1).
- **Deeper, more stable networks.** It is the only method whose loss improves monotonically with strong convergence, up to 512 layers (Section 4.2), and it cuts learning-rate sensitivity by 3–$10\times$ while training stably at rates up to $8\times$ higher than its baseline tolerates (Section 4.3).
- **Beyond language models.** Without tuning, it improves looped latent models on nearly every metric at every scale, and Vision Transformers by up to 0.9 points of top-1 accuracy (Section 4.4).
- **A wider stream, not a narrower one.** LayerRoPE's learned depth-conditioned schedule scales down reads and amplifies writes into the stream, widening the residual stream between 4-$100\times$ its backbone's variance at 1.3B (Section 5.1).



(a) Distribution of MLP block RMSNorm γ values. Means and upper percentiles grow nearly monotonically with depth.



(b) MDS projection of the same per-layer γ vectors. Successive layers trace a near-monotone arc spanning $\sim 143^\circ$.

Figure 2: Norm weights encode depth in LLAMA-like LLMs (LLaMA-2 7B shown here). Magnitude (left) and angular position (right) jointly form a depth-positional code along the residual stream.

2 Norm Weights Encode Depth: A LayerRoPE View

A Transformer exposes exactly one learned gain per layer directly on the residual path: the RMSNorm weights γ. It is therefore the most straightforward site for any depth-conditional code the network may wish to learn. Analyzing pre-trained LLMs, we find that the MLP Norm weights γ carry such a code, expressed jointly through two depth-varying quantities: its *magnitude* and its *direction*.

We observe (see Appendix B) this emergent phenomena is pervasive across LLMs, independent of the model family, architecture or norm formulation. We provide a broad study analyzing 16 pre-trained open-source LLMs *across 9 families, with varying architectures (Dense, Mixture-of-Experts, hybrid linear attention) and norm formulations (Pre/Peri/Post Norm)* in the Appendix B. Below, we consider LLAMA-2 7B for our analysis.

Magnitude grows with depth. Figure 2(a) shows the per-layer distribution of LLaMA-2 7B's MLP block norm weights γ. Its mean and upper percentiles rise near monotonically across layers, with its distribution mean, upper quantiles, showing a clear depth-conditioned trajectory. Note that the *upper bounds of the gamma distributions* themselves are bounded by the respective subsequent layers.
Direction rotates with depth. We perform a multi-dimensional scaling (MDS) projection of the same vectors across layers. Figure 2(b) shows how the γ vectors trace a near-monotone arc of $\sim 143^\circ$ across depth. Successive γ rotate coherently rather than merely rescaling.

Together, magnitude and direction appear to be consistent with a learned depth-positional code: each layer is identified by the length and orientation of its norm weights γ. As we find broadly, this phenomenon is most pronounced in the MLP block's norm (post-attention norm) compared to the Attention block's norms (input norms). We make this analogy explicit in Section 3.

Head-wise Decomposition. Decomposing γ across dimensions for each head of the hidden dimensions (Figure 3) reveals that $\sim 99\%$ of dimensions follow a single near-monotone trajectory; this shared schedule drives the magnitude growth, with the rotation coming from a sparse $\sim 1\%$ of outlier dimensions. The depth encoding is therefore almost *entirely a per-layer scalar depth-conditioning of the same magnitude trajectory* shared across thousands of dimensions, with direction supplied by a handful of outliers. This decomposition motivates our LayerRoPE parameterization (Section 3): replace the bulk per-layer γ with a single shared vector plus a small set of layer-conditioning scalars, and let an explicit complex factor carry the magnitude and rotation that the network would otherwise distribute across L independent vectors. See Appendix C for a more detailed analysis of the head-wise decomposition.

3 LayerRoPE

Intuitively, *LayerRoPE is RoPE along the depth axis*: where RoPE (Su et al., 2024) encodes token position by rotating feature pairs, LayerRoPE encodes the *layer index* by rotating and scaling a global

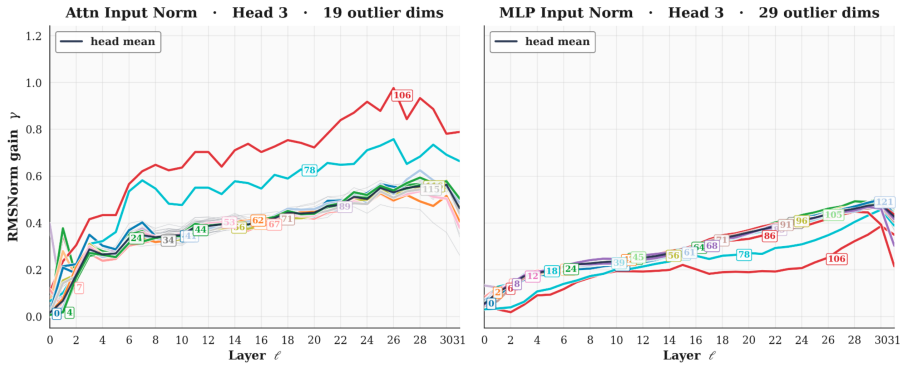


Figure 3: Per-dimension RMSNorm gain γ across layers, for the Attention-input (left panel) and MLP-input (right panel) norms of a particular head (Head 3), in LLaMA-2 7B. Almost all dimensions share a single, near-monotone depth-conditioned trajectory.

per-feature gain γ with depth, to get layer-wise gains γ_ℓ. Applied at the input and output of every block, this explicit depth code regulates how each block reads from and writes to the residual stream, rather than suppressing the stream itself.

Transformer normalization layers commonly apply such a learned per-feature gain γ_ℓ to the normalized activations $\hat{x}$, yielding $\gamma_\ell \odot \hat{x}$, along with an optional additive bias. A standard Pre-Norm transformer stores L such independent γ_ℓ across the L layers; LayerRoPE instead generates every γ_ℓ from a site-specific global shared $\gamma \in \mathbb{R}^d$ and a depth-conditioned complex factor, with one shared γ for the input and output of each block. For each of the four sites (input, output of the Attention, MLP blocks) therefore, one γ is shared across all layers. LayerRoPE additionally conditions the layer-wise γ_ℓ, using just four site-specific scalars $(\alpha^{\text{mag}}, \beta^{\text{mag}})$, $(\alpha^{\text{rot}}, \beta^{\text{rot}})$; a reduction from $O(L\,d)$ per-layer gain parameters, to $O(d)$.

Construction. LayerRoPE multiplies global γ by a single depth-conditioned complex number, with both magnitude and rotational components,

$$\gamma_\ell \;=\; \gamma \odot_c c(\ell), \qquad c(\ell, j) \;=\; \underbrace{e^{r(\ell)}}_{\text{magnitude}}\;\underbrace{e^{i\,\theta(\ell,j)}}_{\text{rotation}} \tag{1}$$

where $\odot_c$ is the complex Hadamard product, $\ell \in [0, L-1]$ the layers index, and j the vector dimension. Each layer's gain is thus the shared γ scaled and rotated by depth conditioned magnitude $e^{r(\ell)}$ and rotation $e^{\theta(\ell,\cdot)}$ terms. Note: $\odot_c$ can be applied by simply reading the real gain $\gamma \in \mathbb{R}^d$, as $d/2$ complex pairs $\tilde{\gamma}_j := \gamma_{2j} + i\,\gamma_{2j+1}$.

Magnitude Dept-conditioning. The *magnitude* schedule scales all pairs equally through a log-linear function of depth, $r(\ell) = \alpha^{\text{mag}} + \beta^{\text{mag}} \log(\ell + 1)$, a single per-layer scalar, matching the shared magnitude trajectory of Section 2.

Rotation Depth-conditioning. The *rotation* schedule spreads a per-layer base angle $\theta(\ell) = \exp\big(\alpha^{\text{rot}} + \beta^{\text{rot}} \log(\ell + 1)\big)$ across pairs by the fixed RoPE spectrum, $\theta(\ell, j) = \theta(\ell)\, b_0^{-2j/d}$, where b_0 is the RoPE analagous base-frequency hyperparameter.

Computational cost. Since LayerRoPE is input-independent and computed once per forward pass, it is quite efficient; Adding no matrix multiplications, and with an at most 0.013% training FLOPs change over Pre-Norm across 60M–1B scales, and effectively none over Peri-Norm. Analytical FLOP estimates are provided in Appendix A. LayerRoPE also replaces the $2Ld$ (Pre-Norm) or $4Ld$ (Peri-Norm) per-layer gain parameters with just $4d + 16$ parameters; causing a net parameter *reduction* compared to both Pre- and Peri-Norm backbones (90K and 188K fewer at 1B).

4 Experiments and Results

We study LayerRoPE across an extensive set of experiments, studying parameter (Section 4.1) and depth scaling (Section 4.2), learning rate robustness (Section 4.3), as well as applying it to latent looped models and vision transformers (Section 4.4). We also carefully ablate and study the learning dynamics induced by LayerRoPE (Section 5).

4.1 Model Scaling Ladder: Scaling LayerRoPE to 100B+ tokens

We begin by applying LayerRoPE to a model scaling ladder from 58M to 1.3B parameters, training every model on C4 at $4\times$ the Chinchilla-optimal budget (Hoffmann et al., 2022), upto 107B tokens.

Setup. We pre-train LLaMA-style (Touvron et al., 2023a) backbones on C4 (Dodge et al., 2021) with the same training data and ordering, and differ only in their specific method: Pre-Norm, Post-Norm, Peri-Norm (Kim et al., 2025), Layer-Norm Scaling (LNS) (Sun et al., 2025), and LayerRoPE on the Pre-Norm backbone. Inspired by Bi et al. (2024), we carefully tune every method by the same procedure: the peak learning rate is swept independently for each method at each scale from 58M to 554M, at the full budget (80 tokens per parameter) until the optimum is bracketed i.e. the loss diverges on both ends of the optimal. Following Bi et al. (2024), extrapolate a per-method power law fit to the 1.3B scale. LayerRoPE's own hyperparameters are held fixed across all scales. See Appendix D.1.2 for details on this learning rate sweep. The tuned baselines are then used to fit a joint power law scaling curve $L(C) = E + A_m C^{-\alpha_m}$, which we report in Figure 1b. We also report downstream evaluations on all the tuned baselines in Table 1. Details of the setup, compute accounting and fit are provided in Appendix D.1; a discussion of the analytical FLOP estimates in Appendix A. Finally, to guard against multiple-seed variability, we ran three seeds of the 58M and 134M scales for all methods at their optimal learning rates, and report these in Appendix D.1.1.

		Accuracy (%, ↑)									
	Method	**ARC-e**	**HellaS.**	**PIQA**	**BoolQ**	**OBQA**	**Wino.**	**LMB.**	**MMLU**	**Avg (↑)**	**LMB PPL (↓)**
134M 10.7 B	Pre-Norm	35.0	31.1	63.9	56.2	**28.8**	48.2	22.6	23.0	40.8	110.5
	Post-Norm	36.2	30.1	62.1	58.8	27.2	51.1	21.1	**23.0**	41.0	125.1
	Peri-Norm	36.6	31.5	63.0	**60.6**	27.8	**51.3**	24.4	22.9	42.2	83.1
	LNS	37.1	32.0	63.2	57.3	27.4	48.8	23.1	22.9	41.3	93.8
	LayerRoPE	**37.8**	**32.6**	**64.1**	59.6	28.6	51.2	**25.9**	23.0	**42.8**	**68.5**
554M 44.3 B	Pre-Norm	39.7	38.8	67.8	59.6	30.0	**54.2**	31.5	23.0	46.0	34.7
	Post-Norm	28.7	26.1	53.6	45.8	25.2	51.3	0.3	22.9	33.0	–†
	Peri-Norm	42.4	42.0	68.3	**59.6**	29.8	52.1	35.3	22.9	47.1	26.1
	LNS	41.6	42.7	68.8	48.7	**31.0**	51.8	34.7	**23.0**	45.6	26.3
	LayerRoPE	**42.8**	**44.9**	**69.5**	56.9	30.6	53.3	**36.1**	22.9	**47.7**	**22.5**
1.3B 107.1 B	Pre-Norm	43.3	46.9	70.2	52.9	31.2	52.4	39.1	23.0	48.0	21.0
	Post-Norm	–	–	–	–	–	–	–	–	–	–
	Peri-Norm	45.0	50.2	72.1	**61.3**	34.0	**55.8**	43.1	23.4	51.6	15.7
	LNS	45.7	51.5	71.9	57.6	32.8	54.5	42.2	22.9	50.9	16.7
	LayerRoPE	**48.1**	**53.8**	**72.3**	58.6	**35.2**	55.5	**44.7**	**23.8**	**52.6**	**14.0**

Table 1: **Downstream Evaluations on scaling ladder.** Zero-shot accuracy of all methods at their tuned learning rate. Within each scale, **best** and second-best are marked per column. † indicates outlier Lambada perplexity ($> 10^4$). (–) indicate Post-Norm failed to converge well at 1.3B scale.

Results. *LayerRoPE reaches Pre-Norm's 1.3B loss with an estimated **3.4x less training compute** (Figure 1b), while using fewer parameters and at most* 0.013% *more training FLOPs (Table 5).* LayerRoPE attains the lowest loss at every scale (Figure 1b), and its margin over Pre-Norm *roughly doubles* from 58M to 1.3B. The joint power-law fit also gives LayerRoPE the steepest compute exponent of all methods (Figure 1b). The gains carry over to downstream tasks (Table 1): LayerRoPE has the lowest Lambada perplexity at every scale and the highest average accuracy at essentially every scale, and at 1.3B it is best on six of the eight tasks. Over multi-seed runs at 58M and 134M scales, LayerRoPE retains the lowest loss under every seed (Appendix D.1.1).

Learning-rate sweeps. Figure 4 shows the learning-rate sweep behind the optimal LR model-ladder: the final validation loss of each run against its peak learning rate, for all five methods at the four swept scales. Because each method is compared at its own tuned optimum, no method is handicapped by a rate tuned for another. We see that: i) LayerRoPE's optimal learning rate *is* 4–$8\times$ *that of Pre-Norm*; ii) its loss is nearly flat over a wide range of rates around that optimum; and iii) it trains stably at rates at which every baseline diverges. The baselines, by contrast, diverge at progressively lower rates as the models grow, and at 554M Post-Norm collapses at all but its smallest rates. This motivates our analysis in Section 4.3 to investigate this learning rate robustness in detail.

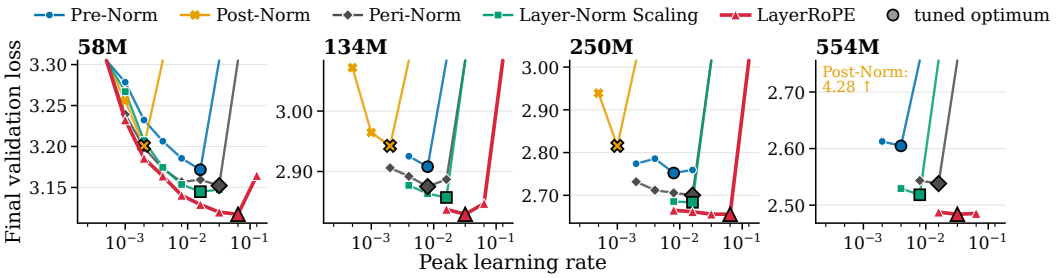


Figure 4: **Model Ladder Learning-rate Sweep.** Each method is swept on the shared $\times 2$ grid until its optimum is interior; outlined markers indicate the tuned optima, whose rates are listed in Table 6.

4.2 Depth Scaling: Scaling LayerRoPE to 512 Layers

Scaling depth is the regime in which the "curse of depth" (Sun et al., 2025) is most severe. To isolate it, we fix a narrow LLaMA-style backbone and vary only the number of layers, $L \in \{48, 96, 192, 256, 384, 512\}$, from 18M to 111M parameters.

Setup. We compare against the standard depth-stabilizing normalizations, including DeepNet (Wang et al., 2024), which was designed to train very deep Transformers. Every model is trained on C4 for upto 20K steps (at least $\sim$1x Chinchilla optimal). First, we train with a shared peak learning rate of 10^{-3}, with all methods trained with two seeds at every depth up to 384 layers; at 512 layers, we report a single seed. These depth scaling results are reported in Figure 5. Similar to the model scaling ladder, we also perform an exhaustive method-specific learning rate sweep, and report the equivalent depth scaling analysis with optimal LR in Appendix D.2.

Results. *LayerRoPE is the only approach whose loss decreases* near monotonically *with strong convergence, upto 512 layers, while also reaching the lowest loss (Figure 5) with a margin that grows with depth.* Every baseline degrades or fails with depth: Post-Norm diverges from 192 layers, Peri-Norm and Layer-Norm Scaling degrade beyond 96 layers, and Pre-Norm stops improving after 256 layers and regresses at 512. DeepNet is the only baseline that improves at every depth, but it remains well above LayerRoPE throughout. From Appendix D.2, where we report an exhaustive learning-rate sweep for all methods at every depth, the equivalent depth scaling curves show that these results are method-specific (and unaffacted by exhaustive baseline tuning): with every method at its own best rate, LayerRoPE still attains the lowest loss at every depth, while Post-Norm, Peri-Norm and Layer-Norm Scaling still break down with depth. *Widening rather than damping the residual stream therefore scales strictly better with depth.*

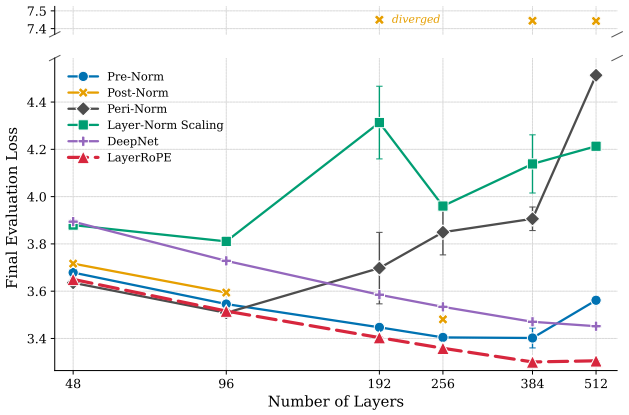


Figure 5: **Depth scaling.** Final validation loss against the number of layers. Points are means over two seeds and bars span both seeds (up to 384 layers). The upper strip shows diverged Post-Norm runs.

4.3 LayerRoPE Widens the Learning-Rate Basin and Delays Divergence

Setup. We sweep the optimal learning rate over two to three orders of magnitude for Pre-Norm and Peri-Norm, each with and without LayerRoPE, at 58M, 368M and 1.3B scales (Figure 6). The 58M 368M and 1.3B models are trained on C4 with each scale sharing the same data, recipe and seed, and LayerRoPE keeping the configuration of Section 4.1 without any tuning. Figure 6 reports the final validation loss for all model scales across this sweep. We also measure the LR sensitivity (Wortsman et al., 2024) metric for LayerRoPE in Appendix D.3.

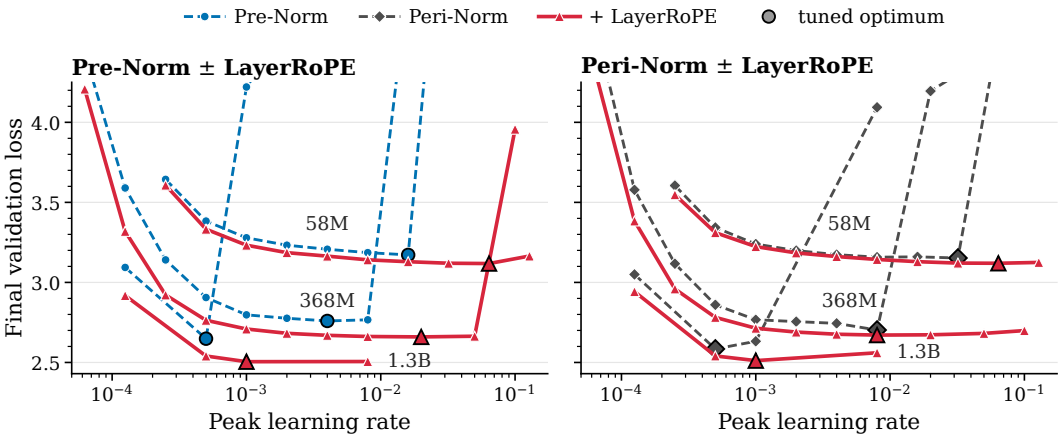


Figure 6: **Learning Rate Basins.** Final validation loss against peak learning rate for Pre-Norm (left) and Peri-Norm (right), each with and without LayerRoPE, at 58M, 358M and 1.3B scales. Runs with losses too high for the panel or diverged, are omitted.

Results. *LayerRoPE promotes a wider basin and later divergence.* Learning-rate robustness is usually characterized in two ways. One, by how quickly the final loss degrades as the rate moves away from its optimum, summarized as LR sensitivity (Wortsman et al., 2024); Two, by the largest rate at which training remains stable, which normalization placement is known to govern (Xiong et al., 2020; Kim et al., 2025). We show that LayerRoPE improves both, on both backbones and at every scale (Figure 6), and it changes the loss vs learning-rate curve in three distinct ways: ***(i) A better optimum.*** LayerRoPE's best loss is lower than its baseline's in all six settings, and in five of them it is reached at a higher learning-rate. ***(ii) A wider basin.*** Around its optimum, LayerRoPE's loss is nearly flat. At 368M on the Pre-Norm backbone, it varies negligibly from 4×10^{-3} to 5×10^{-2} (a $\sim 10\times$ range of learning rates); a range over which Pre-Norm itself diverges. ***(iii) Later divergence.*** LayerRoPE trains stably at rates up to $\sim 8\times$ higher than those at which its baseline first fails, and at 58M and 1.3B on both backbones and at 368M on Peri-Norm it trains at every rate we swept. *Together, these effects reduce LR sensitivity by* $3\times$ *to* $10\times$ *in every setting (Table 8).*

The learning rate sweeps of the scaling ladder extend this picture to all five methods at the full $4\times$ Chinchilla budget (see Figure 4, Appendix D.1.2): the baselines diverge at progressively lower rates as the models grow, whereas LayerRoPE keeps a flat basin at $4\times$ to $8\times$ Pre-Norm's optimal rate at every scale. LayerRoPE appears to consistently improve the sensitivity of Pre- or Peri-Norm backbones near their optimum learning rates, and their stability away from it.

4.4 LayerRoPE for Looped Latent Models and Vision Transformers

We next test whether LayerRoPE transfers out of the box to other architectures and modalities, by applying it without any tuning to looped latent language models and to Vision Transformers.

	Model	PPL↓ Val.	PPL↓ Lambada	Acc.(%)↑ Avg.
140M 11.2 B	GPT	20.28	105.6	29.9
	Parcae	18.66	81.2	33.0
	+ LayerRoPE	**18.36**	**67.3**	**34.9**
370M 29.6 B	GPT	15.01	37.2	40.6
	Parcae	14.22	32.5	43.4
	+ LayerRoPE	**14.05**	**28.0**	**44.0**
770M 61.6 B	GPT	12.80	20.8	47.0
	Parcae	12.49	18.6	50.1
	+ LayerRoPE	**12.09**	**18.4**	**50.6**

Table 2: **Looped latent models.** Validation and Lambada ppl. and downstream accuracy, with looped models at T=8.

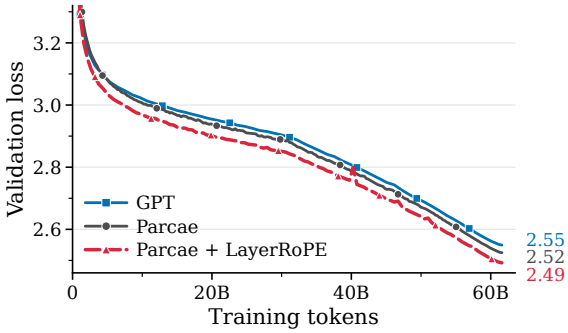


Figure 7: **Looped latent models at 770M.** Validation loss over training, with looped models at T=8; final losses at the right. The loss axis omits the first ∼1B tokens.

Looped latent models. Parcae (Prairie et al., 2026) is a looped language model: a prelude of Transformer layers embeds the input, a recurrent block of layers is applied T times to a latent state, and a coda decodes the result. We train Parcae at 140M, 370M and 770M parameters with its official recipe, on 11.2B, 29.6B and 61.6B tokens of FineWeb-Edu (Penedo et al., 2024), alongside a non-looped GPT baseline with the same parameter count and recipe. We apply LayerRoPE at the input and output of every layer, as in Section 3: each layer of the recurrent block keeps its position in the network as its depth index at every iteration, and the magnitude slope is initialized at zero. Nothing else is changed; the official recipe, including its optimizer settings and stabilization, is used as is, without any tuning for LayerRoPE.

LayerRoPE improves Parcae at every scale (Table 2): Parcae + LayerRoPE is the best model on nearly every metric, with lower validation perplexity, markedly lower Lambada perplexity at 140M and 370M, and higher average downstream accuracy. At 770M, its validation loss stays below both Parcae's and GPT's throughout training (Figure 7), and its advantage over Parcae holds at every number of loop iterations, diminishing as loops are added (see Figure 17). Critically, we observe that Parcae + LayerRoPE's recurrent state grows to norms of $\sim 10^9$ at 770M against Parcae's $\sim 10^3$ (see Figure 18; Appendix D.4), yet its token states remain markedly less collinear for most of training. LayerRoPE thus appears to make fuller use of the recurrent block's capacity, suggesting that a controlled growth in norm can aid, rather than hinder, the use of depth. See Appendix D.4 for details and further analysis.

Vision Transformers. We train both ViT-T and ViT-S (Dosovitskiy, 2020) on ImageNet-1k (Russakovsky et al., 2015) for 300 epochs with the DeiT recipe (Touvron et al., 2021a). Similar to LLMs, we apply LayerRoPE naively at every block, conditioning the gains γ of the pre-norm LayerNorms (while keeping the default layer-wise centering and bias), with the magnitude slope initialized at zero; the recipe is otherwise unchanged. LayerRoPE improves both ViTs on every metric (see Table 3), raising top-1 accuracy by 0.9 points for ViT-T and 0.6 points for ViT-S, at even slightly fewer parameters.

	Acc. (%) ↑		Loss ↓
Model	Top-1	Top-5	Val.
ViT-T	72.21	91.26	1.211
+ LayerRoPE	**73.08**	**91.62**	**1.171**
ViT-S	79.80	94.89	0.897
+ LayerRoPE	**80.38**	**95.26**	**0.865**

Table 3: **ViTs on ImageNet1k.** Each metric is measured as the best over training.

5 Analysis and Ablations

We now examine how LayerRoPE achieves these gains: how it shapes the residual stream across depth (Section 5.1) and which depth schedule it learns (Appendix E). We then ablate its components (Section 5.2).

5.1 Analyzing Variance Growth: LayerRoPE Widens the Residual Stream

We measure the variance of the residual stream after every layer of the 368M and 1.3B models of Section 4.3. We observe that while methods like Layer-Norm Scaling aim to suppress the variance

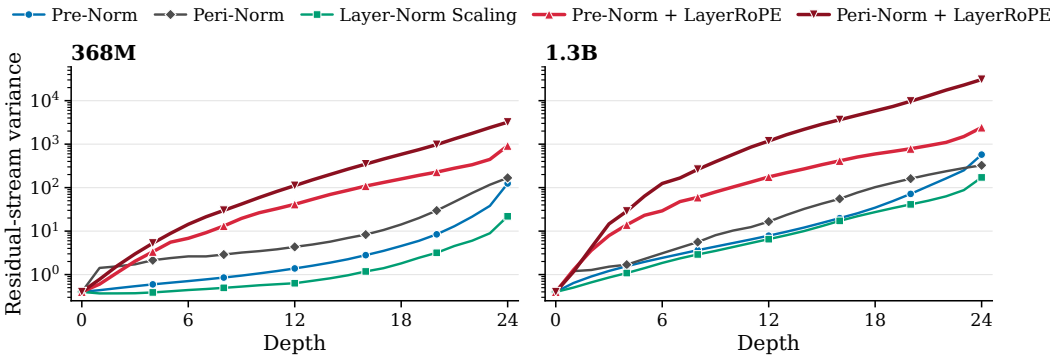


Figure 8: **LayerRoPE widens the residual stream.** Activation variance of the residual stream with model depth, across 368M and 1.3B models, measured at the output of each layer, on the C4 validation set.

growth observed in standard norm forms (Pre-Norm, and even more so in Peri-Norm), LayerRoPE in particular appears to do exactly the opposite and amplifies the variance growth further (Figure 8): at 1.3B, LayerRoPE on the Pre-Norm backbone ends with $4\times$ the variance of Pre-Norm and $14\times$ that of Layer-Norm Scaling, and on the Peri-Norm backbone with $94\times$ that of Peri-Norm. Rather than damping the residual stream, LayerRoPE widens it, attenuating what each block reads while learning to amplify what it writes. See Appendix E for more analysis of the learned depth schedule in LayerRoPE.

5.2 Ablating LayerRoPE's Components

We ablate LayerRoPE at 134M (Table 4). Its magnitude and rotation schedules are complementary: on both backbones, each alone improves on the baseline, magnitude more than rotation, and their combination is best (Table 4a). This mirrors the depth code of Section 2, which is carried jointly by the magnitude and the direction of the gains.

	Peri-Norm		
Variant	**Loss ↓**	**PPL ↓**	**Params**
Baseline	3.0720	21.585	134.12M
LayerRoPE	**3.0252**	**20.599**	134.09M
- Magnitude only	3.0351	20.803	-
- Rotation only	3.0533	21.186	-

(a) Magnitude and rotation schedules.

	Pre-Norm		
Variant	**Loss ↓**	**PPL ↓**	**Params**
Baseline	3.1003	22.205	134.11M
LayerRoPE	**3.0317**	**20.732**	134.09M
- Per-layer γ	3.0318	20.735	134.12M
- Uniform rotation	3.0379	20.862	134.09M

(b) Shared gains and rotation spectrum.

Table 4: **Ablating LayerRoPE at 134M.** Validation loss and perplexity, best in **bold**; Baseline is the backbone without LayerRoPE. (a) Removing either schedule degrades LayerRoPE on Peri-Norm, though each alone still improves on the baseline. (b) Separate gains γ per layer add parameters without improving on LayerRoPE's single shared γ, while one rotation angle for all feature pairs underperforms RoPE's spectrum of frequencies.

Two further design choices hold up (Table 4b). Learning a separate γ for every layer, instead of generating all gains from one shared γ, adds 33.8K parameters without lowering the loss: as the head-wise analysis predicts (Section 2), per-layer gains are largely redundant. Rotating every feature pair by a single angle, instead of by RoPE's spectrum of frequencies, raises perplexity, so the multi-frequency structure like that of RoPE, matters along depth as well.

6 Discussion and Conclusion

Norm weights encode depth. The growth of the residual stream has been read almost universally as a pathology to suppress. Across 16 pre-trained LLMs, we find that it reflects an emergent depth code in the per-layer norm weights, which grow in magnitude and rotate in direction layer by layer, with $\sim$99% of dimensions sharing a single trajectory. LayerRoPE makes this code explicit by applying RoPE along depth rather than along the sequence: every per-layer gain is generated from one shared γ and four scalars per site, with fewer parameters than the gains it replaces.

Regulate the blocks, not the stream. With depth made explicit, LayerRoPE attains the lowest loss at every scale we train, is the only method whose loss keeps improving with strong convergence, up to 512 layers, reduces learning-rate sensitivity by 3–10$\times$, and improves looped models and Vision Transformers without tuning. Its learned schedule inverts the premise behind prior remedies: rather than suppressing the residual stream, LayerRoPE *widens* it, to between $4\times$ to $100\times$ its backbones' variance, by damping what each block reads while amplifying what it writes (Section 5.1). The same preference surfaces elsewhere. Layer-Norm Scaling's learned gains largely undo its fixed down-scaling, and LayerRoPE lets a looped model's recurrent state grow six orders of magnitude beyond Parcae's while its token states stay less collinear for most of training (Appendices D.4 and E). Depth stability, our results suggest, depends less on how large the residual stream grows than on whether each block can scale what it reads and writes to its depth.

Transformers have long been told where each token sits in the sequence. Our results suggest they also benefit from knowing where each layer sits in depth.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and to implement purely supportive code assisting the experiments; the underlying method and its core implementation were developed independently by the authors and manually verified. We have not used generative AI tools to interpret the primary experimental results, develop theoretical models or conceptual frameworks, or determine the scientific claims or conclusions of this work, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to create or edit experimentally supporting (however, not core methodological) code , draft and edit portions of the manuscript, assist in analyzing existing literature and identifying relevant works or gaps, and search for information. We have reviewed all AI-assisted work. All AI-assisted text was manually reviewed, edited, and verified by the authors; literature, references, and externally identified information were manually read through from their original sources. To ensure correctness of the core methodology and implementation, multiple rounds of manual verification of the code was undertaken, as well as experimentation across different pipelines to support validity of all claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

To the best of our knowledge, we know of no significant concerns.

REPRODUCIBILITY STATEMENT

The LayerRoPE construction is fully specified in Section 3, with implementation and computational details provided in the appendices. Complete and ***extensive*** experimental configurations, including model architectures, data and tokenization, optimization and precision settings, training budgets, learning-rate sweeps, evaluation protocols, seed controls, and compute accounting, are documented in the Appendix. We additionally report multi-seed experiments, hold LayerRoPE's configuration fixed across scales, and use established recipes unchanged in our transfer experiments. Code for all major experiments will be released publicly, and we in fact intend to submit LayerRoPE to public, independently verifiable, open-source benchmarks.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pp. 1352–1361. PMLR, 2021.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

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*, pp. 2397–2430. PMLR, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International conference on learning representations*, volume 2024, pp. 2632–2652, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 1286–1305, 2021.

Peijie Dong, Ruibo Fan, Yuechen Tao, Di Mou, Wenhu Hu, Zhenheng Tang, Yinghao Yu, Jiamang Wang, Wenbo Su, Guodong Yang, et al. Dissecting outlier dynamics in llm nvfp4 pretraining. *arXiv preprint arXiv:2602.02047*, 2026.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

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

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.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3392–3405, 2021.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

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.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021a.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 32–42. IEEE, 2021b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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.

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.

Mingze Wang, Shuchen Zhu, Yuxin Fang, Binghui Li, Kai Shen, and Shu Zhong. Negligible in size, significant in effect: On scale vectors in large language models. *arXiv preprint arXiv:2605.26895*, 2026a.

Shaowen Wang, Bingrui Li, Ge Zhang, Wenhao Huang, Shen Yan, and Jian Li. On the residual scaling of looped transformers: Stability and transferability. *arXiv preprint arXiv:2606.18524*, 2026b.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414, 2022.

Mitchell Wortsman, Peter Liu, Lechao Xiao, Katie Everett, Alexander Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations*, volume 2024, pp. 49844–49869, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

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. *arXiv preprint arXiv:2002.04745*, 2020.

Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. *arXiv preprint arXiv:2410.01405*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite depth neural networks. In *International Conference on Learning Representations*, volume 2024, pp. 55099–55150, 2024.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, volume 2025, pp. 97183–97219, 2025.

A Computational Cost Analysis

We provide an analytical FLOP estimate and a parameter comparison of LayerRoPE to that of Pre and Peri-Norm backbones.

Parameter Comparison. LayerRoPE simply replaces the per-layer normalization weights with shared depth-conditioned norm weights that use 16 learned scalars in total (8 scalars each for magnitude and rotation, respectively). Relative to the corresponding backbone, it therefore uses fewer parameters: $2Ld - 4d - 16$ fewer for Pre-Norm and $4Ld - 4d - 16$ fewer for Peri-Norm.

As Table 5 shows, LayerRoPE is slightly smaller than its backbone at every scale (such as 90K and 188K fewer at 1B scale).

Analytical FLOP Estimates. LayerRoPE introduces no matrix multiplications. It's per-layer function at the input and output of the computational block can be generalized as $\tilde{\gamma}_\ell = e^{\alpha+\beta\log(\ell+1)}\, R(\theta_\ell)\,\gamma$ depending only on the layer index, a couple of learned scalars, and a shared d-dimensional γ vector. Since it does not depend on the input, it is therefore *simply computed once per forward pass* and applied as a per-channel scale. Note that the $R(\theta_\ell)$ is the RoPE rotation matrix with layer depth, instead of token position.

Under the standard FLOP accounting, which counts dense matrix multiplications, attention (Kaplan et al., 2020; Hoffmann et al., 2022) among other major operations, but omits secondary normalization and element-wise costs. LayerRoPE, therefore, adds no FLOP increase to the architecture it modifies (Pre or Peri Norm). For a more conservative estimate, we additionally count every elementwise operation introduced by LayerRoPE, assigning one FLOP per scalar addition or multiplication in both the forward and backward passes. At existing normalization layers, $\tilde{\gamma}_\ell$ therefore replaces the existing RMSNorm weight, and adds no per-token cost.

- **For Pre-Norm backbones**, the only new per-token operation is the residual gate $\tilde{\gamma}_\ell \odot f(x)$ on each of the $2L$ sublayer outputs. Each application costs approximately $4d$ training FLOPs: d in the forward pass and $3d$ in the backward pass (one multiplication for the activation gradient, and one multiplication plus one accumulation for the gate gradient). Thus, LayerRoPE adds $8Ld$ FLOPs per training token.
- **For Peri-Norm backbones**, the gate is absorbed into the weight of the output norm that Peri-Norm already applies, so the per-token cost is identical to Peri-Norm.

Following the standard convention (Kaplan et al., 2020; Hoffmann et al., 2022), a LLaMA-style Transformer block with standard MHA and a SwiGLU MLP ($d_{\text{ff}} = \frac{8}{3}d$) costs $72d^2$ matrix-multiplication FLOPs per training token. LayerRoPE's overhead over Pre-Norm is therefore at most $8d/(72d^2) = 1/(9d)$. Under the same elementwise accounting, the two additional RMSNorms of Peri-Norm cost approximately $24Ld$ training FLOPs, corresponding to an upper-bound of $1/(3d)$, three times LayerRoPE's overhead.

			Pre-Norm	FLOPs overhead (%)		LayerRoPE param reduction	
Model	d	L	GFLOPs/token	vs. Pre-Norm	vs. Peri-Norm	vs. Pre-Norm	vs. Peri-Norm
60M	512	8	0.26	0.0128	≈ 0	$\downarrow$6.128K	$\downarrow$14.320K
130M	768	12	0.67	0.0110	≈ 0	$\downarrow$15.344K	$\downarrow$33.776K
250M	896	20	1.36	0.0106	≈ 0	$\downarrow$32.240K	$\downarrow$68.080K
500M	1280	24	3.12	0.0079	≈ 0	$\downarrow$56.304K	$\downarrow$117.744K
1B	2048	24	7.72	0.0051	≈ 0	$\downarrow$90.096K	$\downarrow$188.400K

Table 5: Estimated Analytic training FLOPs of the Pre-Norm baseline, and the relative overhead (%) and parameter reduction (green $\downarrow$) of LayerRoPE over Pre-Norm and Peri-Norm. Over Peri-Norm, LayerRoPE adds only the negligible cost ($< 10^{-4}\%$) of constructing the depth-conditioned γ_ℓ, shown as ≈ 0.

B DO NORM WEIGHTS ENCODE DEPTH ACROSS ALL LLMS?

We check whether this depth-conditioned encoding in the norm weights is a phenomenon observable across LLM families and architectures. We look at *16 pre-trained open-weight LLMs across 9 families*, varying in Pre/Post/Peri-Norm formulations, and architectures (Dense, Mixture-of-Experts, hybrid linear attention, parallel MLP-Attention branches). As in Figure 2, we read the *angle* of each layer's γ from an MDS projection of the per-layer γ vectors, and its *magnitude* from the per-layer distribution of γ values. **Overview:** We start with Pre-Norm LLMs, their MLP Block norms (Appendix B.1) which show clear angular and magnitude depth-conditioning, followed by their Attention Block norms (Appendix B.1). We then study Peri- and Post-Norm LLMs (Appendix B.2) and observe the same depth-conditioning effects as Pre-Norm LLMs.

B.1 PRE-NORM LLMS

We study open-weight Pre-Norm LLMs from seven model families, spanning various architectures. **Dense**: LLaMA-1/2 7B (Touvron et al., 2023a;b), LLaMA-3 8B (Grattafiori et al., 2024), Mistral 7B v0.1 (Jiang et al., 2023), Qwen2/2.5 7B, Qwen3 8B (Yang et al., 2025) and Gemma 7B (Team et al., 2024); **Mixture-of-Experts** (GPT-OSS 20B (Agarwal et al., 2025)), **parallel branches:** Phi-2 2.7B and Pythia 2.8B (Biderman et al., 2023); and **hybrid linear-attention:** Qwen3.5 9B (Team, 2026).

Inferences: Norm weight analysis plots of MLP blocks are shown in Figure 9, and the Attention blocks are in Figure 10:

- **Angular encoding.** Figure 9 shows the MDS (Multi-dimensional scaling) plots with successive layers occupying neighboring angles, and showing a clear depth-ordered trajectory. LLaMA-1, GPT-OSS and Phi-2 for instance, trace a single near-monotone arc from the first layer, as LLaMA-2 does in Figure 2.
- **Magnitude encoding.** Norm weight distributions exhibit the depth conditioning effect as well, with the LLaMA models, Mistral and even Qwen3 reproduce a near monotonic change with network depth (as in LLAMA-2 in Figure 2). Other models with more varying depth-wise patterns, though, such as in Phi-2, where magnitude effects are unclear, the angular effects are particular clear, suggesting that angle and magnitude act as complementary carriers of depth.
- **Outliers.** The effects are not perfect across all Pre-Norm LLMs. Firstly, we can see that in all cases, the norm weights in the Attention Blocks do not show any clear depth conditioning, either angular or magnitude. In the three boxed models Qwen 2.5/3.5 & Gemma 7B, the MLP block norms also do not clearly encode depth. Additionally, there are some general nuances in the overall trend, with the initial and final layers often breaking from pattern or showing more varied distributions.

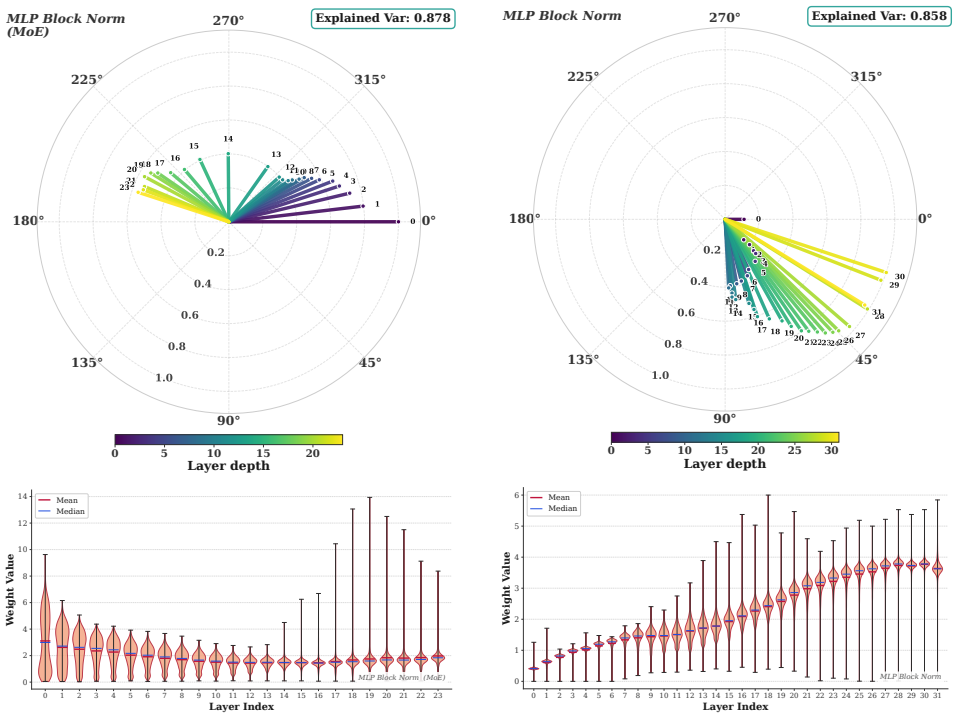
B.2 PERI- AND POST-NORM LLMS

We further study **Peri-Norm** LLMs, which normalize both the input and the output of every block (Gemma 2 9B and GLM-4 9B (Glm et al., 2024)), and **Post-Norm** LLMs, which normalize only the block output (OLMo 2 7B and 13B (OLMo et al., 2024)).[1] By and large, Post-Norm LLMs follow the same trends as Pre-Norm LLMs (Figure 12): their MLP block norms carry a clear depth-conditioned code, while their Attention block norms are only weakly or unclearly conditioned on depth. Interestingly, in Peri-Norm LLMs both the input and the output norms of the MLP block are depth-conditioned (Figure 11), with the effect stronger on the output norms, whose gains grow near-monotonically with depth.

Summary: Across 16 pre-trained LLMs from nine model families, spanning Pre-, Peri- and Post-Norm placements and four architecture types (dense, Mixture-of-Experts, parallel-branch and hybrid linear-attention), the MLP block norm weights by and large exhibit strong depth-conditioning in both magnitude and angle, with only the few exceptions noted in Appendix B.1.

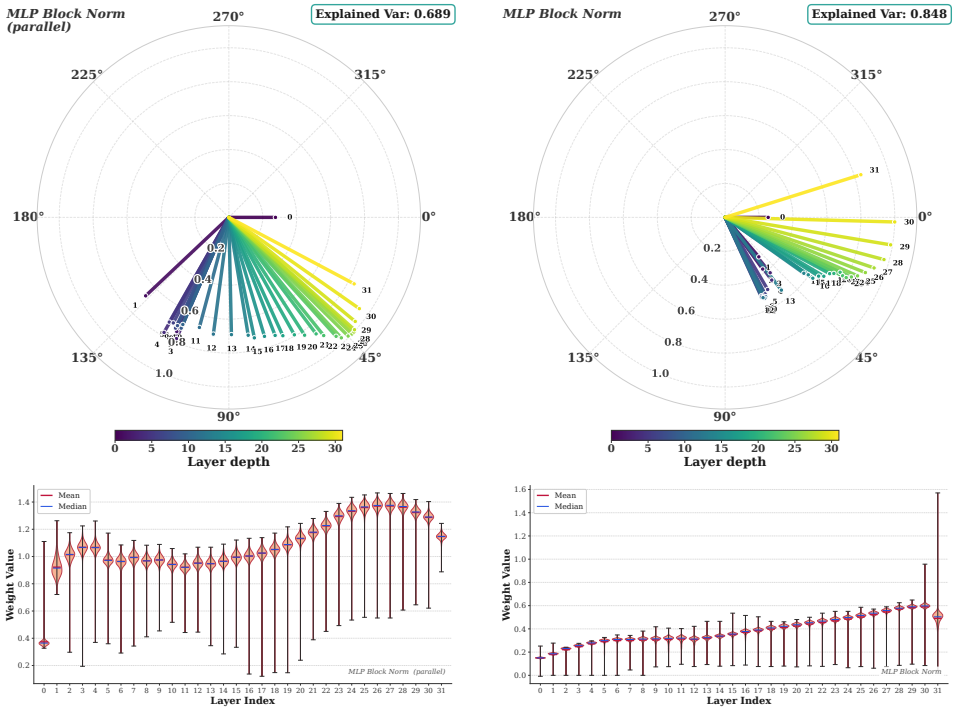
[1] OLMo 2 applies its norm inside the residual branch, $x + \mathrm{Norm}(f(x))$, rather than after the residual addition, $\mathrm{Norm}(x + f(x))$, as in the original Post-Norm Transformer (Vaswani et al., 2017).

Figure 9: **(Pre-Norm, MLP Blocks) Depth-conditioning of Norm Weights.** For each pre-trained model (***GPT-OSS 20B, Mistral 7B, Pythia 2.8B and LLaMA-3 8B***), the distribution of MLP Block RMSNorm γ values and MDS Projection of the same per-layer γ vectors is shown. Note that in all cases, we see strong depth-conditioning in both layer-wise angular and magnitude properties of MLP norm weights.



(a) GPT-OSS 20B

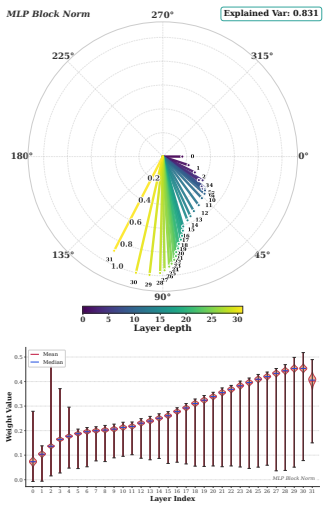
(b) Mistral 7B



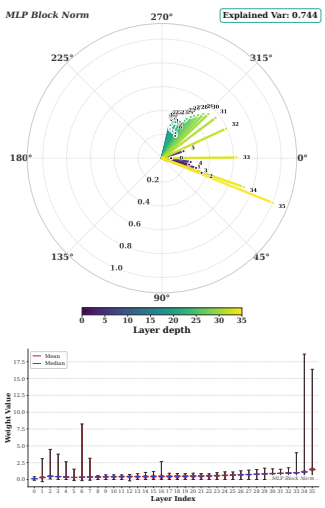
(c) Pythia 2.8B

(d) LLaMA-3 8B

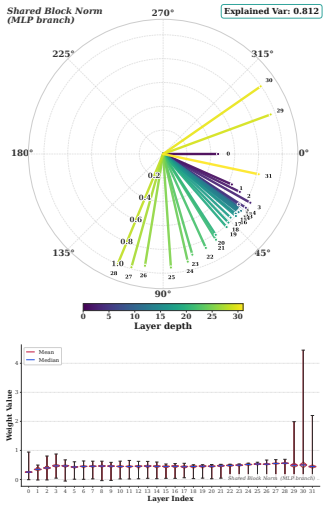
Figure 9: **(Pre-Norm, MLP Blocks. (continued)) Depth-conditioning of Norm Weights.** For each pre-trained model (***LLaMA-1 7B, Qwen3 8B and Phi-2 2.7B***), we continue to see strong depth-conditioning in both layer-wise angular and magnitude properties of the MLP norm weights. We note Qwen2.5, Qwen3.5 and Gemma (highlighted in the boxed row), unlike all other Pre-Norm models, do not show a strong depth conditioning effect in their MLP Norm weights.



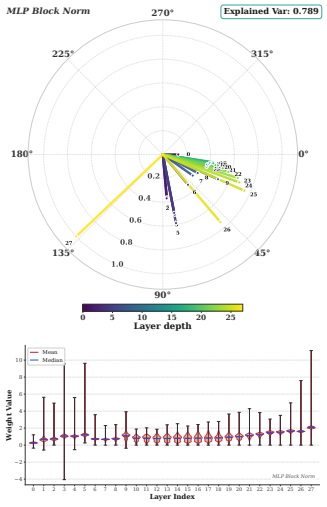
(e) LLaMA-1 7B



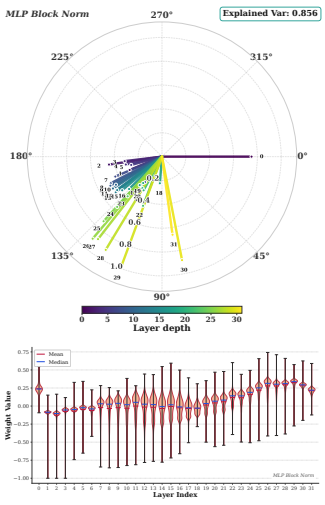
(f) Qwen3 8B



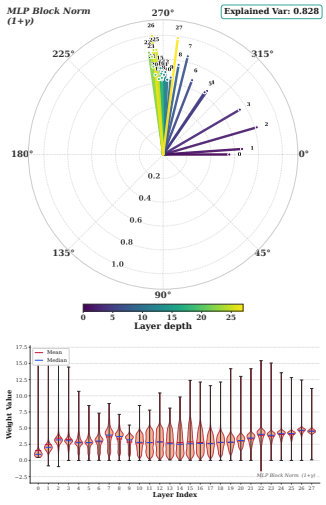
(g) Phi-2 2.7B



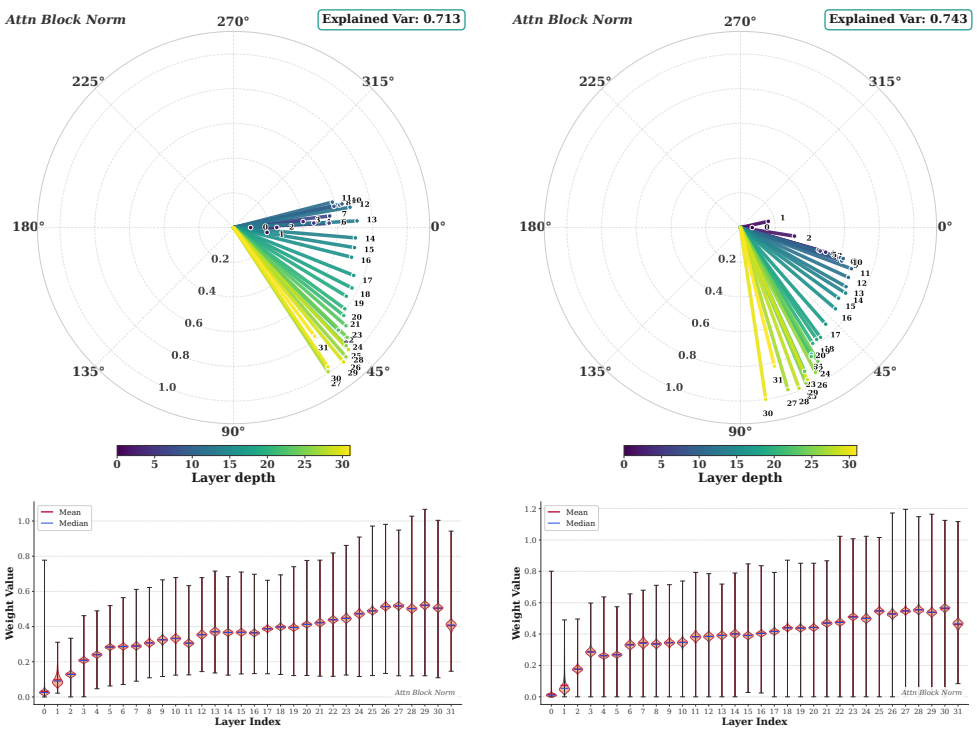
(h) Qwen2.5 7B



(i) Qwen3.5 9B (offsets γ)

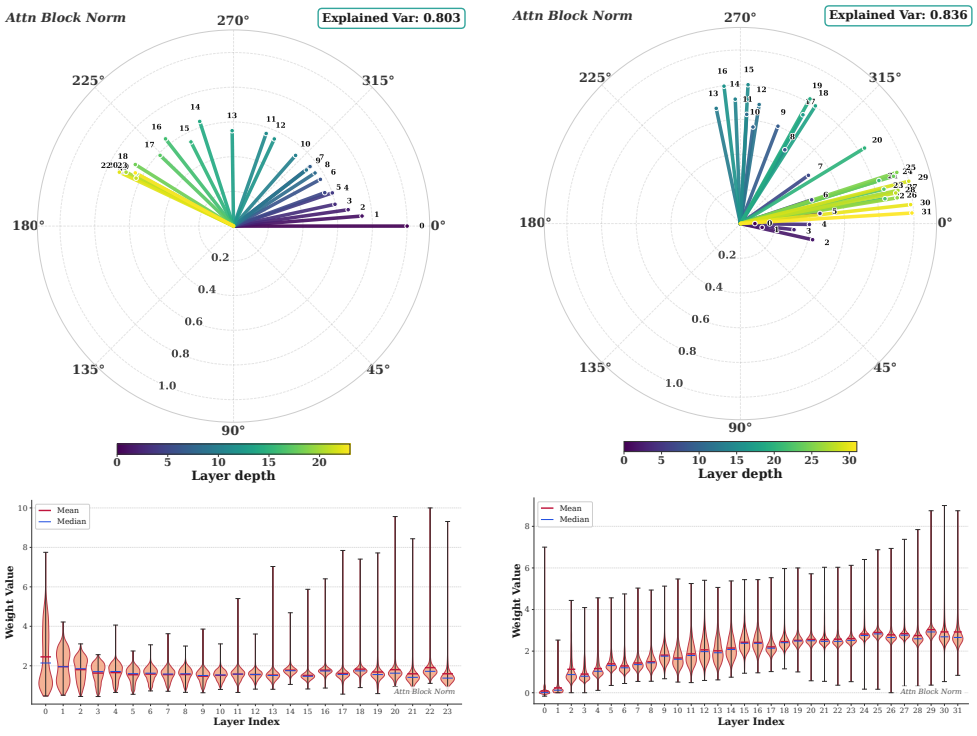


(j) Gemma 7B $(1 + \gamma)$



(a) LLaMA-1 7B

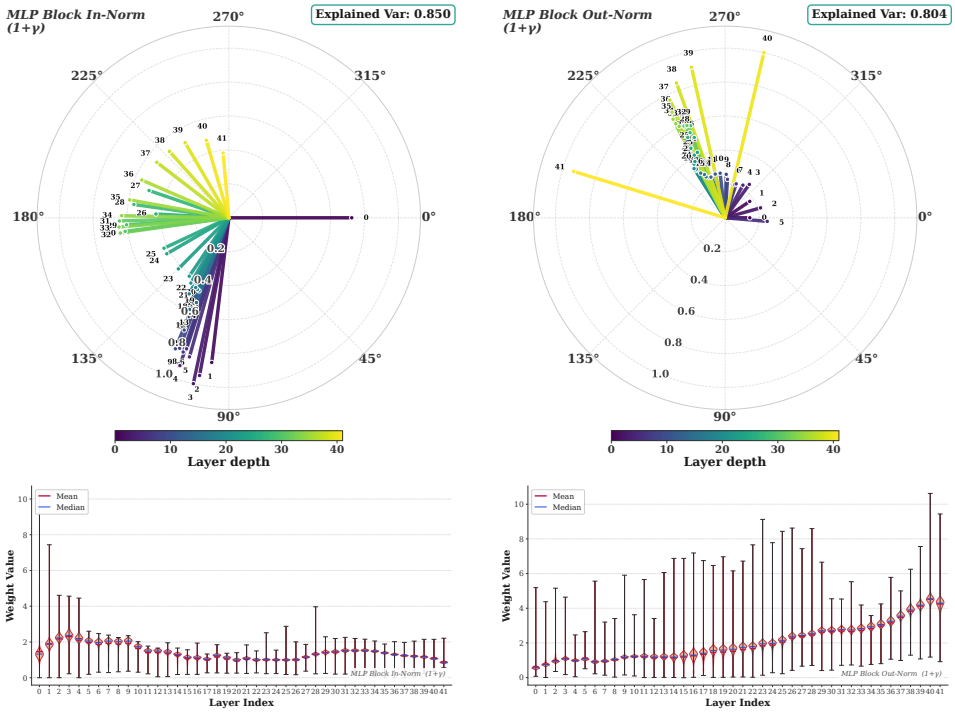
(b) LLaMA-2 7B



(c) GPT-OSS 20B

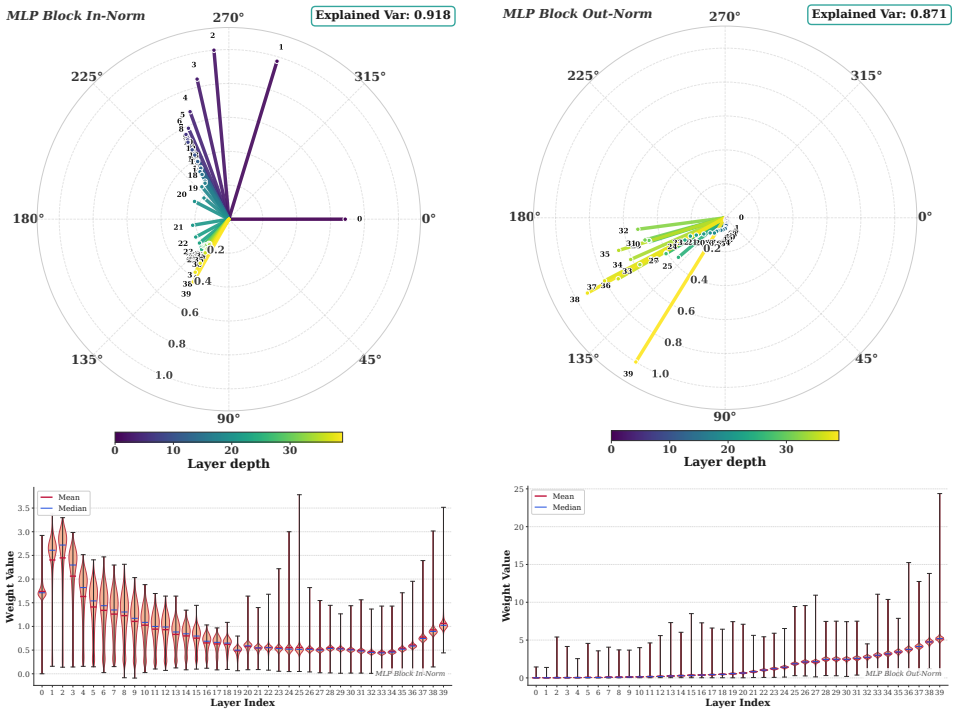
(d) Mistral 7B

Figure 10: **Depth encoding in the Attention block norms of Pre-Norm LLMs.** For each model, *top*: MDS projection of the per-layer Attention block γ vectors (angle); *bottom*: per-layer distribution of the same γ values (magnitude). Compare with the MLP block norms of the same models in Figure 9 (LLaMA-2's in Figure 2).



(a) Gemma 2 9B: MLP input norm

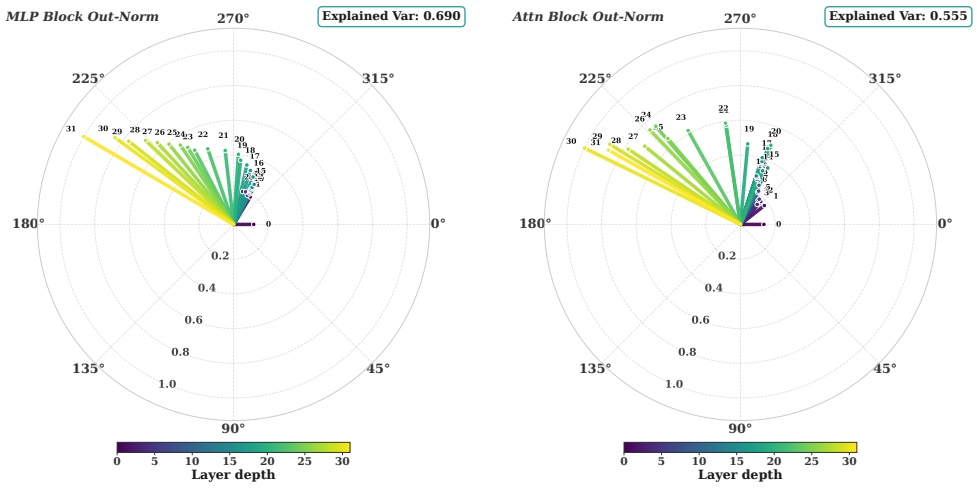
(b) Gemma 2 9B: MLP output norm



(c) GLM-4 9B: MLP input norm

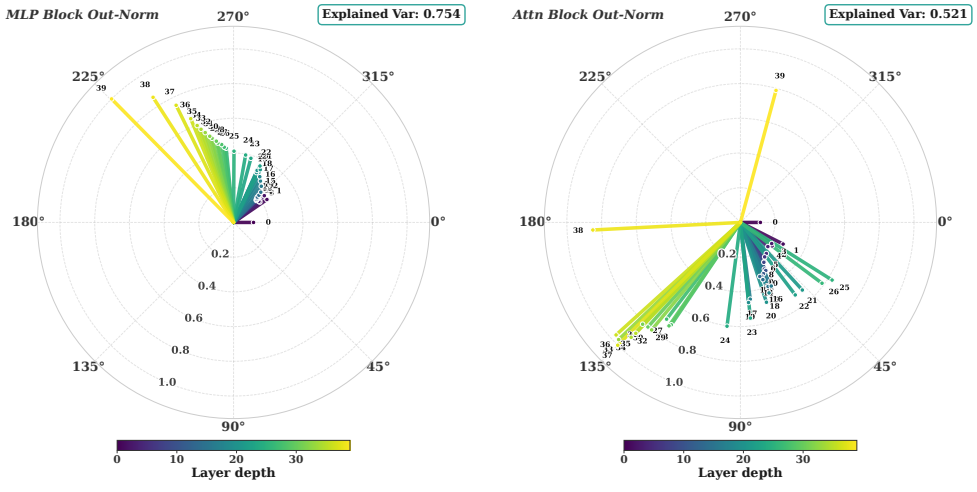
(d) GLM-4 9B: MLP output norm

Figure 11: ***(Peri-Norm, MLP Blocks)* Depth-conditioning of Norm Weights.** For each pre-trained model (**Gemma 2 9B and GLM-4 9B**), the input and output norms of the MLP block are shown: (a, b) Gemma 2 9B and (c, d) GLM-4 9B. In each panel, the MDS projection of the per-layer γ vectors (angle) sits above the distribution of γ values (magnitude); Gemma 2's gains are shown as the applied $1 + \gamma$. We omit the Attention block norms, where as expected, the depth-conditioning is weak



(a) OLMo 2 7B: MLP output norm

(b) OLMo 2 7B: Attention output norm



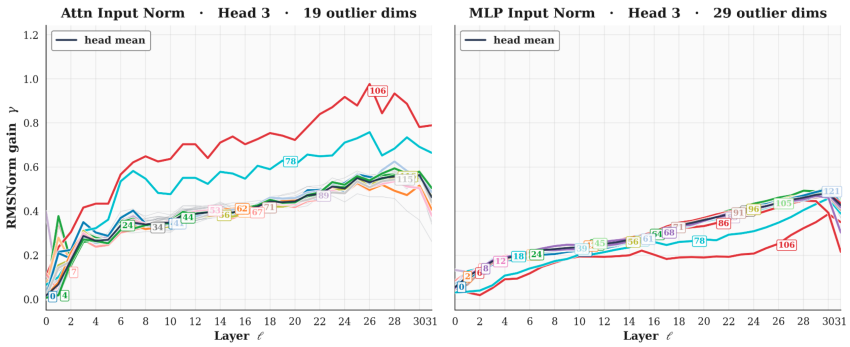
(c) OLMo 2 13B: MLP output norm

(d) OLMo 2 13B: Attention output norm

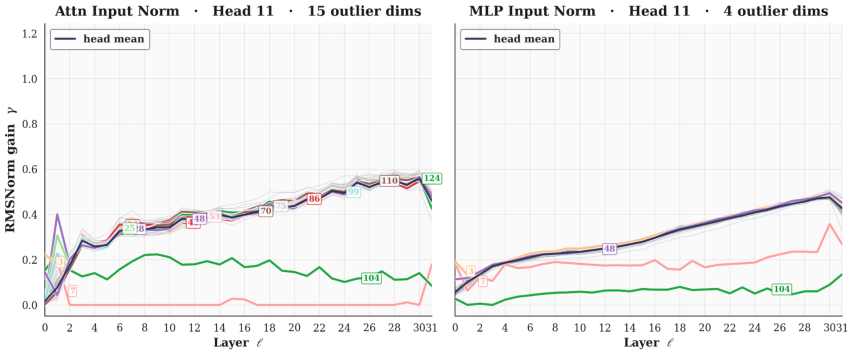
Figure 12: ***(Post-Norm, MLP and Attention Blocks)*** **Depth-conditioning of Norm Weights.** For each pre-trained model (**OLMo 2 7B and OLMo 2 13B**), the MLP and Attention block norms are shown. OLMo 2 normalizes only the output of each block, so each model has an MLP and an Attention output norm. Within each panel, the MDS projection (angle) sits above the distribution of γ values (magnitude).

C HEAD-WISE DECOMPOSITION ANALYSIS OF NORM WEIGHTS

We now investigate the underlying structure behind the observed MDS rotation of γ vectors in Section 2, Figure 2b. We do this by decomposing the per-layer γ vectors of LLaMA-2 7B head-wise, and then track the per dimension growth of each head, over the layers of the network. We follow the head dimensionality of the attention heads, to split the norm weights γ into heads.



(a) Head 3 (Dimensions 384–511 of 4096): Layer-wise growth of Head 3 dimensions of Attention norm weights (left, "Attn Input Norm"), and MLP norm weights (right).



(b) Head 11 (Dimensions 1408–1535 of 4096): Layer-wise growth of Head 11 dimensions of Attention norm weights (left, "Attn Input Norm"), and MLP norm weights (right).

Figure 13: **Per-dimension RMSNorm gain γ across layers** (filtered by specific Heads), for the Attention-input (left panel of each row) and MLP-input (right panel) norms of LLaMA-2 7B.

We observe that the bulk of dimensions ($\sim 99\%$) share one approximate trajectory, with little deviation. For every head we examined, almost all dimensions of γ follow the same near-monotone curve across layers, well captured by the head-mean (Figure 13). *The bulk of the residual-stream depth code is therefore a single scalar trajectory, repeated across thousands of dimensions.* Interestingly, a small percentage ($\sim 1\%$) of outliers, break the trend by a large margin, and deviates strongly. We identify two qualitative classes:

- ***Anti-symmetric outliers.*** The dimension's trajectory in the Attention-input norm is approximately a mirror image of its trajectory in the MLP-input norm (e.g. Head 3 dim 106, Head 11 dim 104). The same scalar dimension is *up-weighted on one branch and down-weighted on the other*, in a layer-dependent way.
- ***Deactivated outliers.*** The dimension's gain is near zero across all layers and both branches (Head 11 dim 7 and dim 104), a consistent null route through the head.

Inferences: This implies that the bulk-mean of the gamma trajectory is essentially a *per-layer scalar magnification of the same trajectory/direction*. The MDS rotation in Section 2 therefore comes from the outliers: the anti-symmetric dimensions reorient the γ vector layer-by-layer, while the deactivated dimensions remain pinned at the origin. This decomposition motivates our LayerRoPE parameterisation: *replace the bulk per-layer γ with a single shared vector plus a small set of layer-conditioning scalars, and let an explicit complex factor carry the magnitude and rotation that the network would otherwise distribute across L independent vectors.*

D EXPERIMENTAL DETAILS

D.1 MODEL LADDER SCALING

We compare methods on a five-model ladder under a common protocol: at every scale, all methods share the data, training recipe and learning-rate tuning procedure, and differ only in normalization.

Models. All models are LLaMA-style decoders (RMSNorm, SwiGLU, RoPE, untied embeddings, head dimension 64), following the setting in Sun et al. (2025), spanning 58M to 1.34B parameters (Table 6). We compare Pre-Norm, Post-Norm, Peri-Norm, Layer-Norm Scaling and LayerRoPE on the Pre-Norm backbone. LayerRoPE uses one configuration at every scale (depth slopes initialized at $\beta = -0.5$ for magnitude and rotation, rotation base frequency 100); like every baseline, only its learning rate is tuned.

Table 6: **Scaling ladder and tuned peak learning rates.** Tokens and C are the total training tokens and compute of each run (Appendix D.1.3). Rates for 60M–500M are measured optima; rates for 1B are single runs at each method's extrapolated optimum (Appendix D.1.2). †Failed to converge.

				Peak learning rate ($\times 10^{-3}$)				
Model	Params	Tokens	C (FLOPs)	Pre	Post	Peri	LNS	LayerRoPE
60M	58.1M	4.6B	1.2×10^{18}	16	2	32	16	64
130M	134.1M	10.7B	7.2×10^{18}	8	2	8	16	32
250M	250.0M	20.0B	2.7×10^{19}	8	1	16	16	64
500M	553.8M	44.3B	1.4×10^{20}	4	0.0625	16	8	32
1B	1.34B	107.1B	8.3×10^{20}	2	0.031†	8	8	32

Data, budget and training recipe. We train on C4 (English) with the LLaMA-2 tokenizer at sequence length 256, padding or truncating each document and masking padding from the loss. This is a single pass over C4 with a batch size of 512, at every scale except 1.3B, which runs for 1.15 epochs to make up the 80 tokens/param. The Adam optimizer is used (β_1=0.9, β_2=0.999, ϵ=10^{-8}) *without weight decay or gradient clipping* for fairness. Learning rate with linear warmup over the first 10% of steps, then cosine decay to $0.1\times$ the peak rate is used. Weights, activations and optimizer states are kept in bf16, except for the normalization gains across all methods, which are kept in fp32 for reliability of the experimentation centered on various normalization formulations. Note, however critically, that in all other experiments for Looped Latent models, ViTs, among others, all parameters are kept in bf16, per the original setting. We report cross-entropy on a fixed 10M-token sample of the C4 validation split. All methods at a scale share one random seed, which fixes both the initialization and data order, so every comparison is paired.

The 1B tier. We noted that, on training the 1B tier above 100B tokens, since the final normalization gain in our experiments were not weight-decayed, the output-logit scale (across all models) kept growing. At 1B it grows large enough that bf16 rounding of the logits destabilized late training. We therefore continued the Pre-Norm, Peri-Norm, Layer-Norm Scaling and LayerRoPE 1B runs with fp32 output logits from a checkpoint that precedes any instability. Post-Norm nevertheless fails to scale: at 500M every rate above 6.25×10^{-5} collapses and the best run plateaus, and at 1B it fails to converge well.

D.1.1 VARIABILITY ACROSS SEEDS.

To measure seed-to-seed variability, we trained every method at 58M and 134M with two further seeds at its tuned rate (Table 7). Standard deviations across seeds are at most 0.016 nats, and LayerRoPE has the lowest loss under every seed at both scales.

D.1.2 LEARNING-RATE SWEEPS

The peak learning rate is tuned per method and per scale on a shared $\times 2$ grid, using the same procedure for every method. At 60M we sweep eight rates (2.5×10^{-4} to 3.2×10^{-2}). At 130M, 250M and 500M we run brackets of four, three and two rates, centered on the method's previous optimum or on its fitted trend. At every scale, the grid is extended until the optimum is interior. A

Table 7: **Seed-to-seed variability on the scaling ladder.** Final validation loss of every method at its tuned learning rate (Table 6), as mean ± standard deviation over three seeds. A seed sets both the initialization and the data order and is shared by all methods, so the last column is a paired difference.

	Final validation loss					Paired gap
Params	Pre	Post	Peri	Layer-Norm Scaling	LayerRoPE	LayerRoPE − Pre
58M	3.172 ± 0.001	3.201 ± 0.001	3.158 ± 0.014	3.147 ± 0.002	**3.129** ± 0.016	−0.043 ± 0.016
134M	2.915 ± 0.014	2.941 ± 0.003	2.884 ± 0.010	2.859 ± 0.002	**2.831** ± 0.001	−0.085 ± 0.014

sweep at 1B was computationally prohibitive, so each method's 1B rate is extrapolated from its four measured optima (Figure 14). We fit $\log \mathrm{LR}^\star = a_m + b_m \log P$ by least squares, evaluate the fit at the 1B model's $P = 1.34$B parameters, and round the prediction to the nearest rate on the $\times 2$ grid (rounding moves no prediction by more than 0.39 grid steps), shown in Figure 14. At 1B, each method is trained once, at the rate given by the fit. Table 6 lists the selected rates.

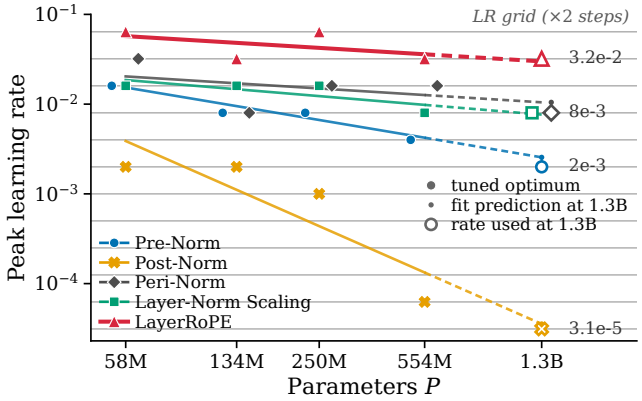


Figure 14: **Extrapolating the tuned learning rate to 1B.** Filled marks are each method's tuned optima at the four swept scales (Figure 4). Lines are the per-method least-squares fits $\log \mathrm{LR}^\star = a_m + b_m \log P$, solid over the fitted range and dashed where extrapolated. At the 1B tier ($P = 1.34$B), each small dot is a fit's prediction and each hollow marker is that prediction rounded to the nearest rate on the $\times 2$ grid (horizontal lines); the hollow markers are the rates used for the 1B runs (Table 6).

D.1.3 Compute Scaling

Training compute is $C = D\,F$, where D counts all training tokens and F is the training FLOPs per token from Table 5 (matrix multiplications and attention, backward pass $= 2\times$ forward). See Appendix A for our Analytical FLOP Estimation of LayerRoPE. We fit $L(C) = E + A_m C^{-\alpha_m}$ by nonlinear least squares to the 20 (method, scale) points. E is shared across methods, since all train on the same data, and (A_m, α_m) are per method; Post-Norm is excluded. The fit gives $E = 1.81$ with an RMS residual of 0.005 nats. The $3.4\times$ compute saving is the ratio of Pre-Norm's 1B compute to the compute at which LayerRoPE's fitted curve reaches Pre-Norm's 1B loss. That loss lies between LayerRoPE's measured 500M and 1B losses, so the saving is an interpolation, not an extrapolation. Because every scale uses the same tokens-per-parameter ratio, these curves compare methods at matched budgets rather than along a compute-optimal frontier.

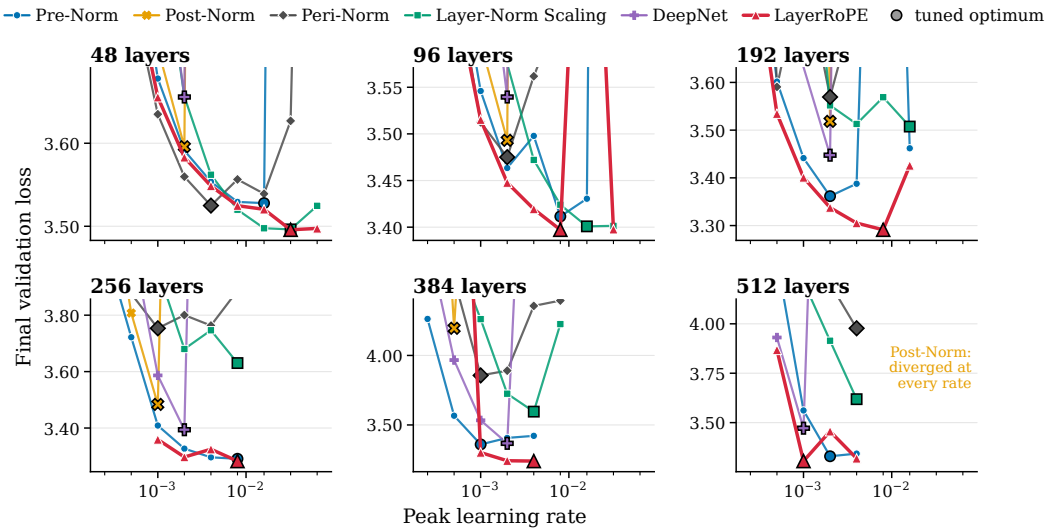


Figure 15: **Learning-rate sweeps at every Depth.** Final validation loss against peak learning rate for the six methods of Figure 5 at each depth. Outlined markers indicate each method's best rate; loss axes are set per panel.

D.2 Model Depth Scaling

Training Configuration. Similar to the model ladder scaling setup, the depth study uses LLaMA-style decoders, with SwiGLU, RoPE and untied embeddings over the 32K-token LLaMA-2 vocabulary. For computational feasibility over 512 layers, we fix the model width to 128 (with SwiGLU feed-forward blocks of width 352 and head size of 64), as we scale depth, from 17.8M parameters at 48 layers to 111.1M parameters at 512 layers. Every model is trained on C4 at sequence length 256 for 20K steps of 512 sequences (2.6B tokens), which is at worst 1x Chinchilla. We follow the optimizer and training recipes from Appendix D.1.

LayerRoPE and Baseline Configuration. LayerRoPE is applied to the Pre-Norm backbone, and using one configuration at every depth. DeepNet (Wang et al., 2024) is applied by scaling the residual stream by a constant α before each post-block normalization and initializes the value, output and feed-forward projections with gain $(8L)^{-1/4}$. The fixed-rate study (Figure 5) uses $\alpha = (2 \times 32)^{1/4}$, the value DeepNet prescribes for 32 layers, at every depth. For completeness, in the below learning-rate sweep, we use the depth-matched $\alpha = (2L)^{1/4}$. We note that in fact, that at the shared rate of 10^{-3}, the fixed-α variant reaches the lower loss at every depth, so this choice does not disadvantage DeepNet. The fixed-rate study reports two seeds per method at every depth up to 384 layers and a single seed at 512 layers.

Learning-rate sweep. To test whether the fixed-rate results reflect the shared rate rather than the methods, we sweep the peak rate at every depth for all six methods with one seed (Figure 15). Figure 16 shows each method's best loss over the sweep. The sweep confirms our prior observations: LayerRoPE attains the lowest loss at every depth. The baselines' failures persist despite their own best tuned learning rates: Peri-Norm and Layer-Norm Scaling degrade beyond 96 layers, and Post-Norm, which trains at its best rate up to 256 layers, degrades sharply at 384 layers and diverges at every rate at 512. This strongly supports our observation of LayerRoPE's depth-scaling benefits.

D.3 Learning-Rate Robustness Sweeps

Setup. The sweeps of Section 4.3 compare Pre-Norm and Peri-Norm, each with and without LayerRoPE, in the configuration of the scaling ladder (Appendix D.1), including LayerRoPE's ($\beta = -0.5$, base frequency 100). The 58M runs are the ladder's own (seed 1). The 368M model has 24 layers of width 1024, and the 1.3B model is the ladder's. Both follow the ladder's recipe with shorter budgets: 60K steps with 6K warmup steps at 368M, and a 100K-step schedule with 1K warmup steps at 1.3B. The rate grid doubles from 2.5×10^{-4} at 58M; at 368M it doubles from 6.25×10^{-5} to 8×10^{-3} and continues with 2×10^{-2}, 5×10^{-2} and 10^{-1}; and at 1.3B it covers rates from 1.25×10^{-4} to 8×10^{-3}.

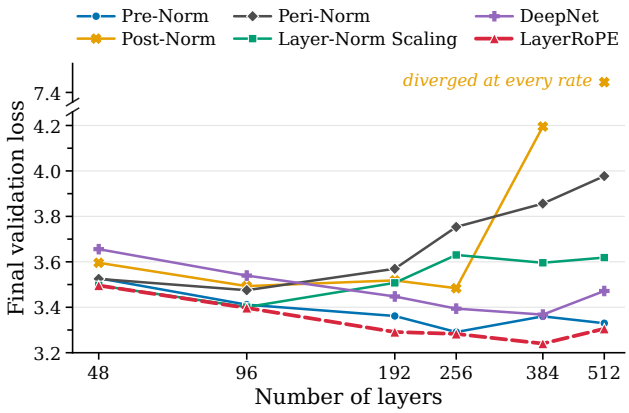


Figure 16: **Depth scaling at the tuned learning rate.** Final validation loss against the number of layers, with every method at its best rate from Figure 15 (seed 1). At 512 layers, Post-Norm diverged at every rate (upper strip).

Metrics. Table 8 summarizes each sweep. Following Wortsman et al. (2024), we measure LR sensitivity as the average of each arm's excess loss over the rates swept for both arms, capping diverged runs at the loss of a uniform prediction. We term a run failed if its final loss exceeds the best loss of its arm by more than one nat; above the optimum, every such run diverged or collapsed. We report the same in Table 8.

Table 8: **Learning-rate robustness in the dense sweeps** (Figure 6), without (−) and with (+) LayerRoPE: the tuned rate $\eta^\star$ and loss $L^\star$, the largest rate that trained stably, and LR sensitivity S (Wortsman et al., 2024), the mean of $\min(L(\eta), L_0) - \min_\eta L(\eta)$ over the rates swept for both arms, with $L_0 = \ln 32000$. Rates are in units of 10^{-3}. ‡Largest rate swept.

		$\eta^\star$		$L^\star$		Largest stable rate		LR sensitivity S	
Backbone	Params	−	+	−	+	−	+	−	+
Pre-Norm	58M	16	64	3.172	3.118	16	128‡	0.458	0.119
	368M	4	20	2.759	2.660	8	50	1.069	0.360
	1.3B	0.5	1	2.648	2.505	0.5	8‡	1.162	0.112
Peri-Norm	58M	32	64	3.152	3.120	32	128‡	0.440	0.087
	368M	8	8	2.702	2.671	8	100‡	1.310	0.263
	1.3B	0.5	1	2.584	2.511	1	8‡	0.506	0.127

Results. We can see from Figure 8, that adding LayerRoPE (+), greatly reduces LR Sensitivity across model scales, and for both Pre- and Peri-Norm, by large margins. This we take to imply a flattening of the loss basin near the optima, and the tolerance to divergence on deviation from the optima as well.

D.4 Looped Latent Models

Setup. We follow the official Parcae recipe (Prairie et al., 2026) at every scale. At 140M, 370M and 770M parameters, the prelude, recurrent block and coda each have 2, 4 and 6 Transformer layers, of width 768, 1024 and 1280, respectively. The GPT baselines are non-looped Transformers of the same width with 6, 12 and 18 layers, identical to Parcae, except the recurrent block iterations. All models here are trained on FineWeb-Edu (Penedo et al., 2024) with Parcae's 32K-token BPE tokenizer, 2,048-token sequences and batches of 256 sequences, for 11.2B, 29.6B and 61.6B tokens, using Muon and AdamW with a trapezoidal learning-rate schedule, gradient clipping at 1.0 and bf16 mixed precision. During training, authors in Prairie et al. (2026) sample the number of loop iterations per sequence. All evaluations are at $T{=}8$ loop iterations, unless stated otherwise. In Parcae + LayerRoPE, LayerRoPE naively generates the RMSNorm gains of every layer from shared

gains through its depth-conditioned magnitude and rotation (Section 3). As mentioned in main text, we naively apply LayerRoPE by using each layer's depth index is its position in the network (independent of the looped iteration). LayerRoPE's magnitude depth slopes are initialized at 0, with rotation base frequency 100. No part of the Parcae recipe is changed or tuned. Downstream, we evaluate on the large tasks (at least 1,000 examples), on which every model performs above chance: ARC-Easy, HellaSwag (zero- and 10-shot), PIQA, Lambada, SQuAD, CoQA and BIG-bench QA WikiData, averaging over three few-shot sampling seeds. The full results are provided here in Table 9.

Loop iterations. Figure 17 evaluates both 770M models at every number of loop iterations. LayerRoPE improves Parcae's convergence at every T, albeit with diminishing benefit as loops are added.

Recurrent state. Figure 18 shows that LayerRoPE lets the recurrent state grow by several orders of magnitude during training, reaching norms of $\sim 10^9$ at 770M against $\sim 10^3$ for Parcae. Yet for most of training its token states stay markedly less correlated than Parcae's, which are nearly collinear (mean pairwise cosine similarity 0.96–0.99). From the markedly less token correlation in the recurrent states, we infer that LayerRoPE encourages better utilization of the recurrent block's capacity during looping. The fact that this occurs while the recurrent state norms are increasing orders of magnitude over vanilla Parcae, is indicative of our finding that controlled norm/variance increase may infact be beneficial to the utilization of depth in Transformers.

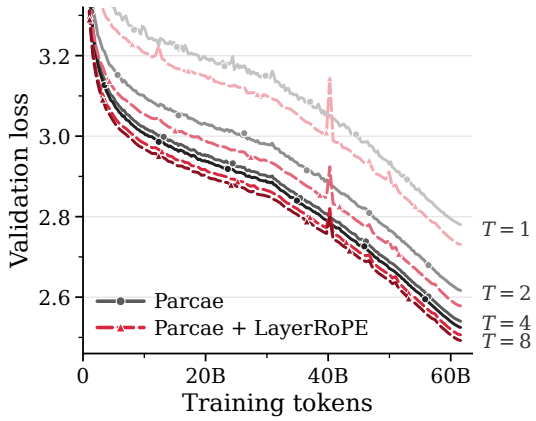


Figure 17: **Every loop depth at 770M.** Validation loss throughout training for Parcae (solid) and Parcae + LayerRoPE (dashed), evaluated at $T \in \{1, 2, 4, 8\}$ loop iterations, from lighter to darker. The loss axis omits the first $\sim$1B tokens.

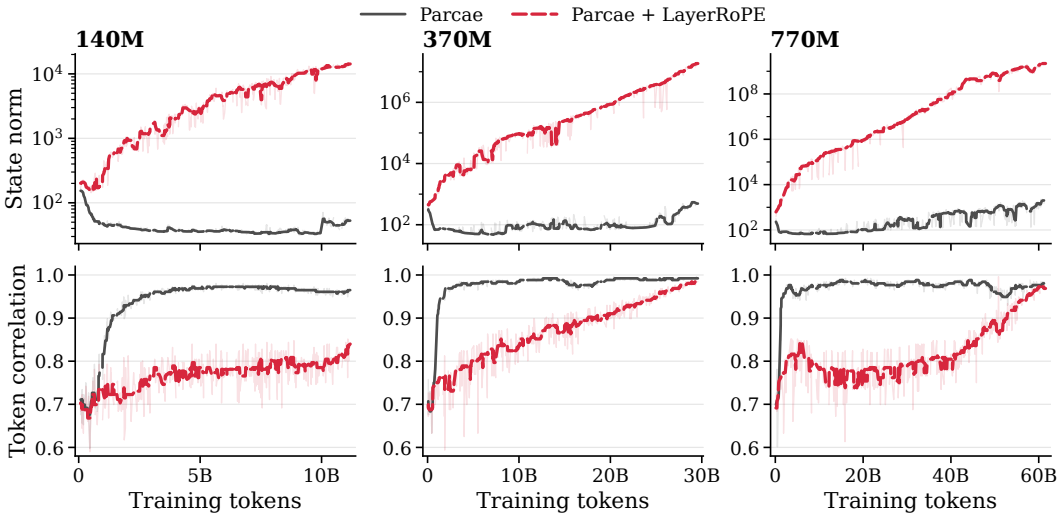


Figure 18: **Recurrent state during training.** Mean norm of the recurrent state leaving the recurrent block (top, log scale) and mean pairwise cosine similarity between its centred token states (bottom), for Parcae and Parcae + LayerRoPE. Lines are rolling medians over logged steps, with raw values shown faintly.

E Learned Depth-Conditioning in LayerRoPE

Setup. We analyze the 368M, 1.3B scale Pre- and Per-Norm models of Section 4.3, with and without LayerRoPE, as well as a Layer-Norm Scaling model trained identically. We pass 512

	Model	Looped Iters. (T)	Val. PPL (↓)	Lambada PPL (↓)	Accuracy (%, ↑) ARC-e	HellaS.-0	HellaS.-10	PIQA	LMB.	SQuAD	CoQA	QA-WD	Avg. (↑)
140M 11.2 B	GPT	–	20.28	105.61	51.0	31.6	31.7	62.2	24.3	5.0	9.8	23.3	29.9
	Parcae	8	18.66	81.17	53.6	33.7	33.7	63.1	26.4	7.5	12.8	33.1	33.0
	Parcae + LayerRoPE	8	**18.36**	**67.32**	**54.0**	**34.3**	**34.5**	**63.9**	**27.1**	**14.0**	**15.6**	**35.7**	**34.9**
370M 29.6 B	GPT	–	15.01	37.16	59.5	40.3	40.1	67.0	32.9	20.2	20.1	44.4	40.6
	Parcae	8	14.22	32.46	**61.4**	43.6	43.5	68.5	35.0	28.0	20.6	46.8	43.4
	Parcae + LayerRoPE	8	**14.05**	**27.98**	61.2	**43.8**	**44.3**	**68.8**	**37.2**	**28.5**	**21.0**	**47.4**	**44.0**
770M 61.6 B	GPT	–	12.80	20.80	64.7	47.8	47.6	69.4	40.5	32.1	24.0	50.3	47.0
	Parcae	8	12.49	18.56	**67.4**	51.2	51.9	**71.2**	42.2	38.0	26.0	52.6	50.1
	Parcae + LayerRoPE	8	**12.09**	**18.44**	65.7	**51.8**	**52.9**	**71.2**	**42.9**	**40.9**	**26.2**	**53.1**	**50.6**

Table 9: **Per-task downstream results of the looped latent models.** Validation perplexity on FineWeb-Edu, Lambada perplexity and accuracy (%) on each downstream task, averaged over three few-shot sampling seeds, with looped models at $T{=}8$; training tokens under each size, best per size in **bold**.

samples from the C4 validation set, and measure the variance of the residual stream after each layer's last residual addition (critically, excluding the first (BOS) position whose massive activations may otherwise dominate the early layers). For the 1.3B scale, we also read out LayerRoPE's learned depth code (Section 3) at its four sites, the input (read) and residual write (out) of the attention and MLP blocks: the magnitude multiplier $e^{r(\ell)}$ and the rotation angle $\theta(\ell)$, whose slopes β^{mag} and β^{rot} are both initialized at -0.5.

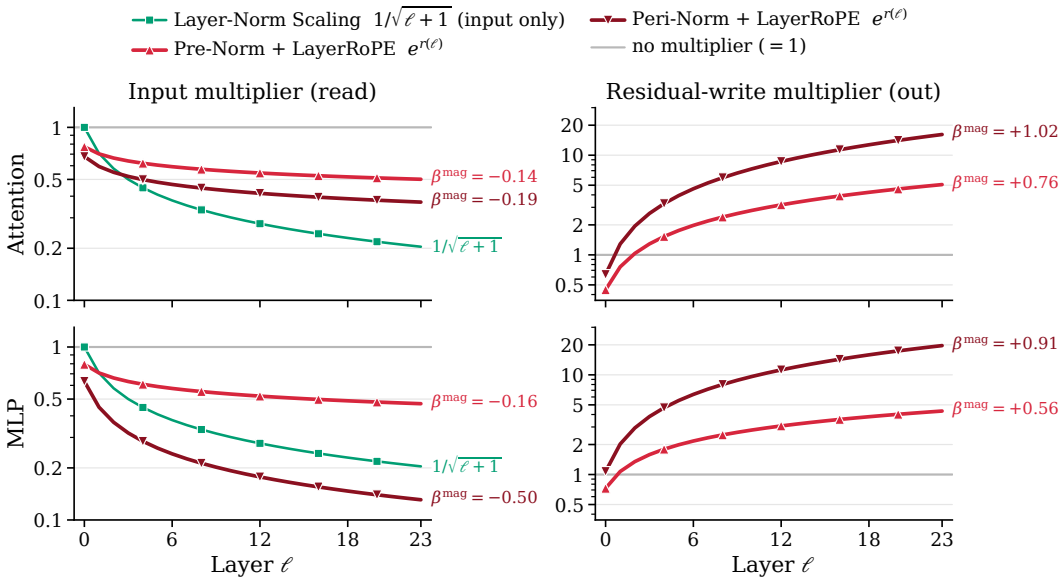


Figure 19: **Learned depth multipliers at 1.3B.** Learned magnitude multipliers $e^{r(\ell)}$ of LayerRoPE against the fixed $1/\sqrt{\ell+1}$ of Layer-Norm Scaling.

Learned multipliers and rotations. Figure 19 contrasts the magnitude multipliers that LayerRoPE learns at 1.3B with the fixed $1/\sqrt{\ell+1}$ input multiplier of Layer-Norm Scaling, and Figure 20 shows its learned rotations. In both backbones we notice that LayerRoPE keeps attenuating what each block reads, but also learns to amplify what it writes. The residual-write slopes, initialized at -0.5 like the input ones, turn positive during training, so that the ***write multiplier grows*** 6–$25\times$ ***from the first layer to the last***.

Effective depth gain. Per-layer normalization weights can encode depth as well, so Figure 21 compares the effective normalization weights γ_ℓ^{eff} of every method, with its depth-dependent factors folded in, as the magnitude ratio $\|\gamma_\ell^{\text{eff}}\|/\|\gamma_0^{\text{eff}}\|$ to the first layer's. With free per-layer weights, Pre-Norm grows its input weights to about $2\times$ the first layer's by the last layer, while Peri-Norm keeps its input weights within the same range, and grows those of its output normalization to upto $\sim 3\times$. Strikingly, we observe that the learned weights of Layer-Norm Scaling largely compensate for their prescribed down-scaling! Note that the ratio never falls below ~ 0.75 at the attention input, and traces a U-shape at the MLP input. LayerRoPE, whose weights are shared across layers, in contrast

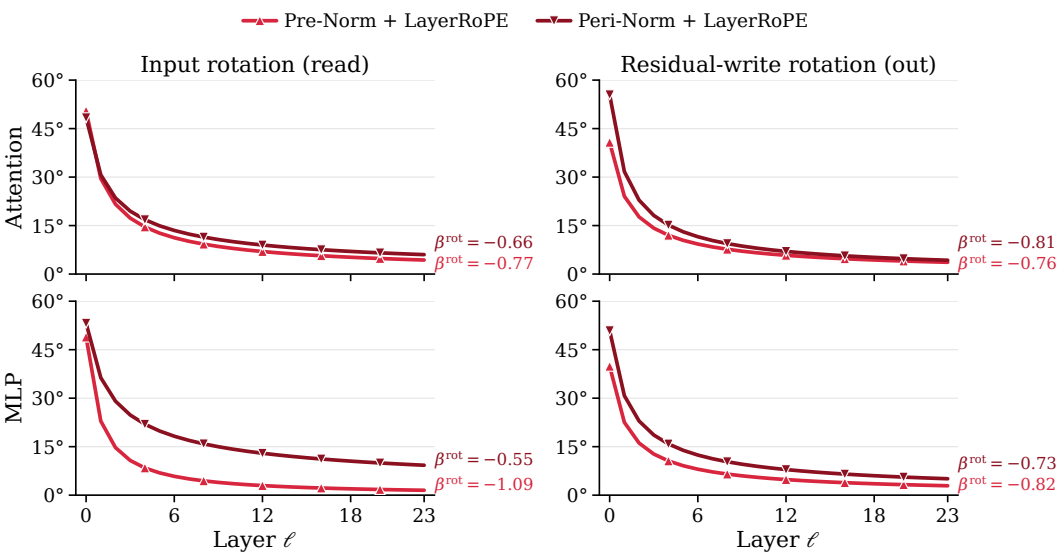


Figure 20: **Learned rotation angles at 1.3B.** Rotation $\theta(\ell)$ of the highest-frequency channel pair at each LayerRoPE site. Labels give the learned slopes β^{rot}.

to all, instead retains a smooth, monotone ratio of $(\ell+1)^{\beta^{\text{mag}}}$. It is unclear though, to what degree this smoothness is beneficial, and we leave this for future work.

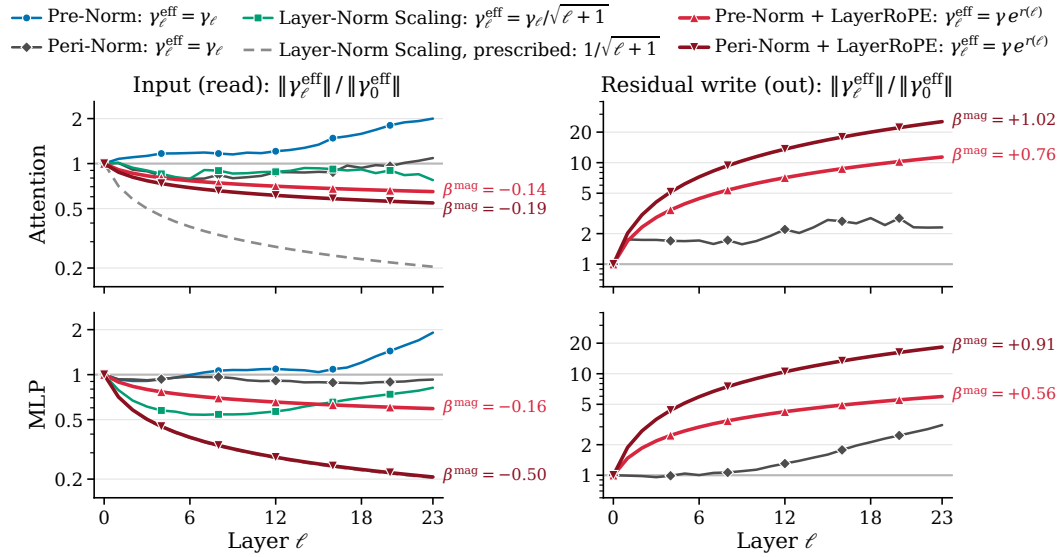


Figure 21: **Effective depth gain at 1.3B.** Magnitude ratio $\|\gamma_\ell^{\text{eff}}\|/\|\gamma_0^{\text{eff}}\|$ of each layer's effective normalization weights to the first layer's, with every depth-dependent factor folded into γ_ℓ^{eff} (legend). For Peri-Norm, the right column shows its output normalization; Pre-Norm and Layer-Norm Scaling have no residual-write weights. LayerRoPE's weights γ are shared across layers, so its ratio is $(\ell+1)^{\beta^{\text{mag}}}$; labels give the learned slopes β^{mag}.

F Related Work

Controlling residual growth. Hidden-state norms/variance grow with depth is a well known phenomena in Pre-Norm Transformers (Sun et al., 2025). Existing work to contain this growth, can be seen to be along two lines: The first *attenuates* what enters or exits the residual stream: Layer-Norm Scaling (Sun et al., 2025) scales each block's input (the output of its normalization) by $1/\sqrt{\ell+1}$, Peri-Norm (Kim et al., 2025) re-normalizes each block's output, while DeepNet (Wang et al., 2024) and Depth-μP (Yang et al., 2024) bound each block's update to train at extreme depth. Others like ReZero (Bachlechner et al., 2021) and LayerScale (Touvron et al., 2021b) gate each residual branch with learned multipliers initialized near zero. The Second line of work *restructures* the residual pathway, by circumventing the single stream problem entirely: Attention Residuals (Team et al., 2026) for instance, replaces the residual with a learned and expensive bounded aggregation of streams over depth, and Hyper-Connections (Zhu et al., 2025; Xie et al., 2025) widen the single stream into several parallel streams, where constrained mixing can preserve signal magnitude. Concurrently, some works like Wang et al. (2026a) have shown that the scale vectors of normalization layers, though negligible in size, provides useful preconditioning of future layers, and Wang et al. (2026b) derive a formulaic depth- and loop-dependent residual scale for looped Transformers. Both treat these gains as fixed or as free per-layer parameters. LayerRoPE is orthogonal to both lines: it directly tackles the singular additive stream problem, by counterintuiviely widening the stream further, generating per-layer gains that regulate each computational block's I/O using one depth-conditioned shared vector.

Outliers and norm weights. A complementary line of work focuses on understanding *where* these activations blow up (and aim to identify *why*). Works on massive activations (Sun et al., 2024), attention sinks (Xiao et al., 2023) and high-norm register tokens (Darcet et al., 2024), show that extreme values concentrate in a few tokens and channels. Many have pointed to normalization gains as having a role: Wei et al. (2022) suggest they amplify outlier channels. Kovaleva et al. (2021) show that norm gains carry high-magnitude dimensions at the same position in every layer, and Dong et al. (2026) show they vary in magnitude with depth and attention type. These works isolate individual outliers or exploit cross-layer redundancy after training. We instead study the systemic trajectory of *all* dimensions of γ across depth: we show that its magnitude and direction evolve jointly, that $\sim 99\%$ of each head's dimensions share a single depth-wise trajectory, and that a sparse set of outlier dimensions cause this rotation effect (Section 2). Finally, LayerRoPE directly builds this structure into the architecture rather than attenuating it.

Positional encodings along sequence and depth. Self-attention is permutation-equivariant, so Transformers encode token order explicitly, originally with sinusoidal or learned vectors added to the input (Vaswani et al., 2017). Rotary position embeddings (RoPE; Su et al., 2024) instead rotate each pair of query and key features by an angle proportional to the token's position, with frequencies on a geometric spectrum, so that attention scores depend on relative position alone. Depth, by contrast, is rarely encoded explicitly: Universal Transformers add a timestep embedding at each recurrent step (Dehghani et al., 2018), and looped Transformers benefit from per-loop scales conditioned on a timestep encoding (Xu & Sato, 2024). We find that standard LLMs learn such a depth code implicitly, in the magnitude and direction of their per-layer gains. LayerRoPE makes it explicit, applying RoPE's rotation, together with a learned magnitude, along depth to a shared gain vector rather than along the sequence to queries and keys.