GeoNorm: Unify Pre-Norm and Post-Norm with Geodesic Optimization

Anonymous ACL submission

Abstract

The placement of normalization layers, specifically Pre-Norm and Post-Norm, remains an open question in Transformer architecture design. In this work, we rethink these approaches through the lens of manifold optimization, interpreting the outputs of the Feed-Forward Network (FFN) and attention layers as update directions in optimization. Building on this perspective, we introduce GeoNorm, a novel method that replaces standard normalization with geodesic updates on the manifold. Furthermore, analogous to learning rate schedules, we propose a layer-wise update decay for the FFN and attention components. Comprehensive experiments demonstrate that GeoNorm consistently outperforms existing normalization methods in Transformer models. Crucially, GeoNorm can be seamlessly integrated into standard Transformer architectures, achieving performance improvements with negligible additional computational cost.

1 Introduction

Recent years have witnessed remarkable progress in Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023), driven primarily by the exponential growth of training data and model parameters. This advancement is reflected in substantial improvements across language modeling (Fedus et al., 2022; Puigcerver et al., 2024; Jiang et al., 2024; Meta; Liu et al., 2024; Team et al., 2025) and computer vision (Riquelme et al., 2021; Lin et al., 2024). Within the Transformer architecture, the choice of normalization function, such as Pre-Norm and Post-Norm, is critical, as it influences training stability, model performance, and gradient flow.

Currently, Pre-Norm has become the predominant choice in modern LLMs due to its favorable stability properties (Liu et al., 2024; Yang et al., 2025; Jiang et al., 2024). In contrast, Post-Norm (Popel and Bojar, 2018; Shazeer and Stern, 2018), which applies normalization after the addition of the residual connection, was originally introduced in the Transformer and facilitates the training of deeper networks. However, it is often prone to training instability and loss spikes (Wang et al., 2024). Pre-Norm was subsequently proposed to improve stability by applying layer normalization to the residual branch before the addition operation (Nguyen and Salazar, 2019). Nonetheless, Pre-Norm introduces imbalanced gradients, with lower layers tending to receive larger updates than higher ones (Wang et al., 2024). To address these limitations, recent methods have sought to enhance stability and enable extreme model scaling. DeepNorm (Wang et al., 2024) stabilizes training and improves performance through better parameter initialization and residual scaling. Meanwhile, SandwichNorm (Ding et al., 2021; Yin et al., 2025) introduces additional normalization layers at the network's input and output. While these approaches demonstrate strong empirical results, they generally lack a unified theoretical framework for analyzing normalization functions in Transformers.

In this work, we interpret the operations $\boldsymbol{x} + \mathrm{FFN}(\boldsymbol{x})$ and $x + \mathrm{Attention}(\boldsymbol{x})$ as optimization steps on a sphere, where $\mathrm{FFN}(\boldsymbol{x})$ and $\mathrm{Attention}(\boldsymbol{x})$ serve as estimated gradient directions produced by the respective modules. From this perspective, each Transformer layer can be viewed as performing an iterative optimization step within a dynamical system: token embeddings act as the initial state, while the attention and feed-forward modules generate update directions based on the current iterate. The normalization operator then projects the updated vector back onto a constraint set, specifically, the sphere defined by the normalization radius. Building on this formulation, our contributions are threefold:

- **Theoretical Framework:** We introduce a

theoretical framework for analyzing normalization in Transformers. Here, $\boldsymbol{x}$ is treated as the current point, while $\mathrm{FFN}(\boldsymbol{x})$ and $\mathrm{Attention}(\boldsymbol{x})$ are interpreted as the update direction in optimization.

- **GeoNorm Normalization:** We propose GeoNorm, a novel normalization function that replaces conventional schemes by leveraging geodesic and Riemannian optimization on the sphere, leading to improved performance.
- **Extensive Empirical Validation:** Through comprehensive experiments across varying training lengths, datasets, and model sizes, we validate the effectiveness of GeoNorm. Additionally, we demonstrate its scalability and generalization on large-scale datasets with downstream task evaluation.

2 Related Work

Normalization Function The original Post-Norm architecture, first introduced in the Transformer model (Popel and Bojar, 2018; Shazeer and Stern, 2018), applies layer normalization to the sum of the identity term and the residual output. While Post-Norm facilitates the training of deep networks, it can suffer from instability and exhibit sharp loss spikes (Wang et al., 2024). In response, the Pre-Norm variant (Nguyen and Salazar, 2019) modifies the residual connection by applying layer normalization directly to the identity term before the addition, leading to improved training stability. However, this often results in imbalanced gradient magnitudes, with lower layers receiving larger gradients compared to higher layers (Wang et al., 2024). To further enhance stability and enable scaling to significantly greater depths, DeepNorm (Wang et al., 2024) introduces refined parameter initialization and residual scaling. Another approach, StandwichNorm (Ding et al., 2021; Yin et al., 2025), incorporates normalization layers at both the network input and output alongside the identity connection. Although these methods have empirically demonstrated strong performance, they currently lack a comprehensive theoretical framework for analyzing the underlying normalization mechanisms.

Gradient Descent. The stochastic gradient descent (SGD) (Bottou, 1998) is an iterative optimization method with the gradient. Based on the SGD, there is SGD with momentum (Rumelhart et al., 1986) which use the momentum of previous gradient. Adagrad (adaptive gradient algorithm) (Duchi et al., 2011) is a modified stochastic gradient descent algorithm with per-parameter learning rate. The RMSProp (Graves, 2013) divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. Adam (Kingma and Ba, 2017) run averages with exponential forgetting of both the gradients and the second moments of the gradients are used.

Optimization Method. The Geodesic optimization is related to the optimization on the manifold (Kingma and Ba, 2017). Riemannian Steepest Descent Method extends the classic gradient descent to optimize functions on curved spaces (manifolds) (Debye, 1909) by following the path of steepest decrease, using the Riemannian gradient (the projection of the standard gradient onto the manifold) and a retraction to map steps back onto the manifold. Riemannian Newton's method (da Silva et al., 2025) adapts classical Newton's method for optimization problems on curved spaces (Riemannian manifolds). The Riemannian Conjugate Gradient method (Hestenes et al., 1952) extends the conjugate gradient method to a manifold. For the retraction relation method, they will calculate the gradient and map the result back to the manifold. The geodesic-based methods (Zhang and Sra, 2016) use the gradient descent method on the geodesic of the manifold.

3 Method

3.1 Post-Normalization in LLMs

Beyond the well-recognized attention and feed-forward network modules, modern LLMs rely critically on normalization to stabilize training and inference as depth increases. Let $\boldsymbol{x}_k$ denote the token representation after the k-th Transformer layer. A Transformer layer with post-normalization can be written as

$$\begin{aligned} \tilde{\boldsymbol{x}}_k &= \mathrm{Norm}(\boldsymbol{x}_k + \mathrm{Attn}(\boldsymbol{x}_k)), \\ \boldsymbol{x}_{k+1} &= \mathrm{Norm}(\tilde{\boldsymbol{x}}_k + \mathrm{FFN}(\tilde{\boldsymbol{x}}_k)), \end{aligned} \tag{1}$$

where $\mathrm{Attn}(\cdot)$ and $\mathrm{FFN}(\cdot)$ denote the attention and feed-forward modules, and $\mathrm{Norm}(\cdot)$ is the normalization operator (typically RMSNorm followed by rescaling to preserve the vector norm).

From this formulation, we may view the Transformer layer as performing an iterative optimization step in a dynamical system. The token embeddings serve as the initial state, while the attention

and feed-forward modules produce update directions based on the current iterate $\boldsymbol{x}_k$. The normalization operator then projects the updated vector back to a constraint set, specifically, the sphere defined by the normalization radius $\|\boldsymbol{x}_k\|$. This interpretation closely mirrors the projected optimization scheme:

$$\boldsymbol{x}_{k+1} = \mathrm{Proj}_{\boldsymbol{\Omega}}(\boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k), \tag{2}$$

where $\boldsymbol{s}_k$ is an update step determined by past iterates $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k\}$, α_k represents the step size, and $\mathrm{Proj}_{\boldsymbol{\Omega}}(\cdot)$ denotes projection onto the feasible region $\boldsymbol{\Omega}$. Equation (2) includes a wide range of classical optimization algorithms such as projected gradient descent.

From this perspective, (1) corresponds to the special case where the feasible region $\boldsymbol{\Omega}$ is the sphere in the Euclidean space induced by ℓ_2-norm, and the projection $\mathrm{Proj}_{\boldsymbol{\Omega}}(\cdot)$ is simply vector normalization. Thus, each Transformer layer applies an unconstrained update followed by a projection back to the sphere, aligning post-normalization Transformers with the general class of optimization algorithms that incorporate projection to enforce constraints. This connection motivates replacing the projection step with a more geometrically informed operation, such as the exponential mapping, in the subsequent section.

3.2 Geodesic-Inspired Transformer Design

Geodesic-based optimization methods are known to possess several structural advantages over projected gradient approaches, particularly when the feasible region $\boldsymbol{\Omega}$ is a smooth manifold (e.g., spheres, Stiefel/orthogonal matrices, low-rank matrices, or fixed-rank positive semi-definite matrices). Traditional projected methods handle constraints by alternating between Euclidean optimization and projection back onto the feasible set. While simple to implement and interpret, this extrinsic approach discards geometric information that the projection step can significantly distort step $\boldsymbol{s}_k$, introduce unexpected kinks in the optimization trajectory, and degrade convergence due to repeated violations and corrections of feasibility. These issues are amplified when the feasible region is a smooth manifold, where the curvature causes projected steps to deviate substantially from the true directions. In contrast, geodesic-based optimization circumvents these limitations by performing updates intrinsically on the manifold. By following geodesics or retraction paths, each iterate remains feasible without projection, and the search direction corresponds to the Riemannian step under the manifold metric. As a consequence, geodesic-based methods typically exhibit smoother optimization trajectories, improved conditioning, and stronger convergence guarantees compared to their projected counterparts. These observations motivate our modification of the Transformer model from projection-based schemes in (1) to a geodesic-based formulation.

In geodesic optimization, the core idea is to modify the step direction $\boldsymbol{s}_k$ in a manner consistent with the geometry of the manifold $\boldsymbol{\Omega}$, ensuring that every iterate remains on $\boldsymbol{\Omega}$. This is achieved through the exponential map $\exp_{\boldsymbol{x}}(\cdot)$. Consider a smooth manifold $\boldsymbol{\Omega}$ with tangent space $T_{\boldsymbol{x}}\boldsymbol{\Omega} := \{\boldsymbol{v} : \boldsymbol{x}^\top \boldsymbol{v} = 0\}$ at point $\boldsymbol{x}$. For any tangent vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\boldsymbol{\Omega}$, there exists a unique geodesic $\gamma_{\boldsymbol{v}}(t)$ satisfying

$$\gamma_{\boldsymbol{v}}(0) = \boldsymbol{x}, \quad \gamma'_{\boldsymbol{v}}(0) = \boldsymbol{v}.$$

The exponential map at $\boldsymbol{x}$ is then defined as $\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \gamma_{\boldsymbol{v}}(1)$. We refer readers to Riemannian optimization literatures (Zhang and Sra, 2016; da Silva et al., 2025) for formal definitions and properties of exponential mappings. For a general manifold $\boldsymbol{\Omega}$, the exponential map is implicit and rarely available in a closed form, which limits the practical adoption of geodesic-based methods due to the computational cost of evaluating $\exp_{\boldsymbol{x}}(\boldsymbol{v})$. However, in our setting (1), the manifold is the sphere, where the exponential map is explicit and computationally efficient.

Let $\boldsymbol{\Omega}$ be a sphere embedded in Euclidean space with the standard induced Riemannian metric. Given a point $\boldsymbol{x} \in \boldsymbol{\Omega}$ and a tangent vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\boldsymbol{\Omega}$, the exponential map admits the closed-form expression

$$\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \cos\left(\frac{\|\boldsymbol{v}\|}{\|\boldsymbol{x}\|}\right)\boldsymbol{x} + \|\boldsymbol{x}\| \sin\left(\frac{\|\boldsymbol{v}\|}{\|\boldsymbol{x}\|}\right)\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}. \tag{3}$$

Given an update direction $\boldsymbol{s}$ for $\boldsymbol{x}$, we first project it to the tangent space:

$$\boldsymbol{v} = \boldsymbol{s} - \frac{\boldsymbol{x}^\top \boldsymbol{s}}{\|\boldsymbol{x}\|^2}\boldsymbol{x},$$

and then update the representation intrinsically via the exponential map $\boldsymbol{x} + \exp_{\boldsymbol{x}}(\boldsymbol{v})$. Specifically, (2) is modified to

$$\begin{aligned} \boldsymbol{v}_k &= \boldsymbol{s}_k - \frac{\boldsymbol{x}_k^\top \boldsymbol{s}_k}{\|\boldsymbol{x}_k\|^2}\boldsymbol{x}_k, \\ \boldsymbol{x}_{k+1} &= \exp_{\boldsymbol{x}_k}(\alpha_k \boldsymbol{v}_k). \end{aligned} \tag{4}$$

Exponential mapping interprets the update as moving along the geodesic determined by the update direction, producing a smooth, norm-preserving, and curvature-informed transformation between states.

These observations motivate us to replace the projection-based normalization in (1) with a geodesic-inspired normalization based on the exponential map (3). More specifically, we reformulate the post-normalized Transformer layer (1) as

$$\begin{aligned} \boldsymbol{s}_k &= \mathrm{Attn}(\boldsymbol{x}_k), \\ \boldsymbol{v}_k &= \boldsymbol{s}_k - \frac{\boldsymbol{x}_k^\top \boldsymbol{s}_k}{\|\boldsymbol{x}_k\|^2}\boldsymbol{x}_k, \\ \tilde{\boldsymbol{x}}_k &= \exp_{\boldsymbol{x}_k}(\alpha_k \boldsymbol{v}_k), \\ \tilde{\boldsymbol{s}}_k &= \mathrm{FFN}(\tilde{\boldsymbol{x}}_k), \\ \tilde{\boldsymbol{v}}_k &= \tilde{\boldsymbol{s}}_k - \frac{\tilde{\boldsymbol{x}}_k^\top \tilde{\boldsymbol{s}}_k}{\|\tilde{\boldsymbol{x}}_k\|^2}\tilde{\boldsymbol{x}}_k, \\ \boldsymbol{x}_{k+1} &= \exp_{\tilde{\boldsymbol{x}}_k}(\alpha_k \tilde{\boldsymbol{v}}_k), \end{aligned} \tag{5}$$

where the update directions $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_k$ are produced by the attention and feed-forward modules, $\boldsymbol{v}_k$ and $\tilde{\boldsymbol{v}}_k$ are their tangent-space projections, and $\{\alpha_k\}$ is a prescribed sequence of step sizes. In contrast to the standard post-normalization, which abruptly projects the updated representation back to the sphere, the update in (5) moves intrinsically along the geodesic determined by the residual direction.

Step-size selection is an important component of geodesic optimization. Classical choices include constant step sizes for geodesically convex and smooth objectives ($\alpha_k \leq \frac{2}{L}$, where L is the gradient Lipschitz constant), polynomially diminishing step sizes for stochastic Riemannian optimization, and Barzilai–Borwein (BB) step sizes for accelerated variants. Since the token representations in Transformer layers evolve under stochastic transformations, arising from dropout, token generation sampling, autoregressive randomness, and the non-deterministic interaction of multiple tokens, we model the layerwise dynamics as a stochastic Riemannian optimization process. Accordingly, we adopt a polynomially decaying step size of the form $\alpha_k = \frac{\alpha}{k^{0.5}}$ (named as **sqrt**), which theoretically achieves the $O(\frac{\log k}{k})$ convergence rate for stochastic Riemannian gradient methods (da Silva et al., 2025). Also we could also have $\alpha_k = \frac{\alpha}{k}$ (named as **harmonic**) and $\alpha_k = \frac{\alpha(T-K)}{T}$ (named as **linear**, and T is the total layer number).

In the proposed method (5), we keep the attention and FFN modules unchanged and modify only the post-normalization step. Instead of applying a projection via the normalization operator $\mathrm{Norm}(\cdot)$, we adopt the exponential map inspired by geodesic Riemannian optimization. Our intuition is that projection-based normalization may destroy certain geometric structure in the update directions $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_k$ generated by the attention and FFN modules. Projection forces the representation back onto the sphere by discarding the radial component abruptly, potentially altering the intended direction of motion. In contrast, the exponential map preserves the intrinsic geometry of the sphere: it transports the tangent vector faithfully along the geodesic without the directional distortions introduced by renormalization. This allows the model to follow the spherical manifold structure naturally, maintaining richer geometric relationships between the current representation $\boldsymbol{x}_k$, the attention update $\boldsymbol{s}_k$, the FFN update $\tilde{\boldsymbol{s}}_k$, and the curvature of the underlying feature space. Comparing the classical Transformer update (1) with our geodesic formulation (5), we note that the exponential map adds negligible computational overhead. Its closed-form expression involves only basic vector operations and introduces no additional trainable parameters. Consequently, our method integrates seamlessly into existing LLM architectures without modifying the core Transformer blocks, and is computationally efficient.

3.3 Pre-Norm with GeoNorm formulation

Moreover, we find that Pre-Norm can be regarded as a special case of GeoNorm (as shown in Appendix C), as it follows an update scheme similar to (5), with the corresponding mapping in (3) but with non-adaptive angles. This interpretation is further supported by empirical results, where the two formulations exhibit comparable performance.

In summary, we interpret Transformers as Riemannian flows of token embeddings evolving on a spherical manifold. Replacing projection with geodesic-informed exponential updates in (5) enables representations to move smoothly along circles on the sphere, leading to more coherent and curvature-aware information propagation across layers.

4 Experiment

Baseline. We compare the proposed GeoNorm with the Dense model and normalization. To be specific, we evaluate GeoNorm against normalization methods, including Pre-Norm, Post-Norm,

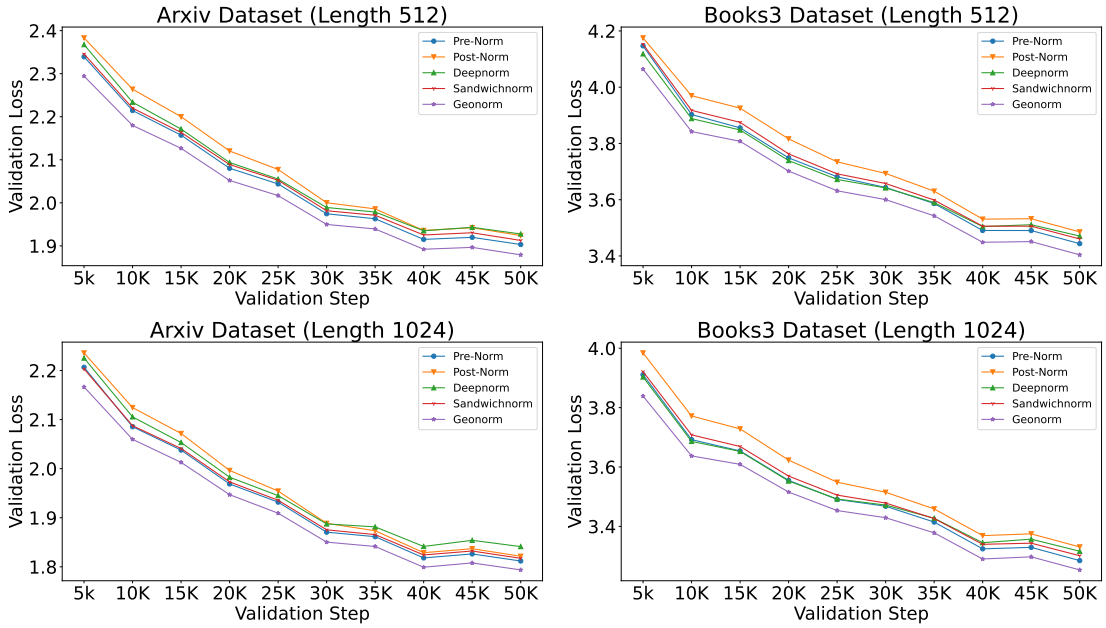

Figure 1: The performance of different methods on the Arxiv and Books3 dataset, with model parameter 125M.

DeepNorm and SandwichNorm. **The implementation of GeoNorm is in Appendix F.**

Datasets. Our analysis involves training language models on the Arxiv and Books3 datasets, which are frequently used benchmarks for evaluating model performance. Moreover, we train the model on the large-scale dataset FinWeb-Edu (Lozhkov et al., 2024) and evaluate on downstream datasets,

Experiment settings. Initially, we compare GeoNorm with other baselines at training lengths 512 and 1024, using decoder-only Transformers (Brown et al., 2020) with model size 125M, whose configuration is shown in Appendix E. Subsequently, we evaluate the performance of larger model sizes, specifically 350M and 1.3 B.

4.1 Compare with baseline

GeoNorm achieves better performance, with different datasets. As shown in Figure 1, to evaluate the generalization capability of the proposed method, we conducted experiments across two distinct datasets, Arxiv and Books3, using a fixed training length of 512. The results demonstrate a clear and consistent performance hierarchy. On the Arxiv dataset, GeoNorm establishes a strong lead with a final loss of 1.8792. This outperforms the closest baseline, Pre-Norm (1.9032), as well as Post-Norm (1.9240), DeepNorm (1.9277), and SandwichNorm (1.9126). Crucially, this pattern of superiority is not an artifact of a single data distribution. It is robustly replicated on the Books3 dataset, where GeoNorm again achieves the optimal loss of 3.4040, maintaining a decisive margin over all alternative normalization methods. The fact that GeoNorm secures the top position on both datasets, which represent different textual domains and complexities, provides compelling evidence that its architectural advantage is not dataset-specific. Therefore, we conclude that GeoNorm offers generalizable improvements, delivering reliably better performance across diverse data environments.

GeoNorm achieves better performance, with different training lengths. The performance of GeoNorm demonstrates consistent superiority across varying training lengths on the Books3 dataset. At a training length of 512, GeoNorm establishes an immediate lead, achieving an optimal loss of 3.4040. This already surpasses the results of all comparative baselines, including Pre-Norm (3.4437), Post-Norm (3.4855), DeepNorm (3.4708), and SandwichNorm (3.4604). Crucially, this initial advantage is not merely maintained but is significantly amplified when the training duration is extended. At the longer training length of 1024, GeoNorm again records the best possible loss of 3.2534. More importantly, the performance gap separating it from the nearest competitor widens considerably at this scale; while Pre-Norm achieves a loss of 3.2847, the other methods fall even further behind. Therefore, the superiority of GeoNorm is

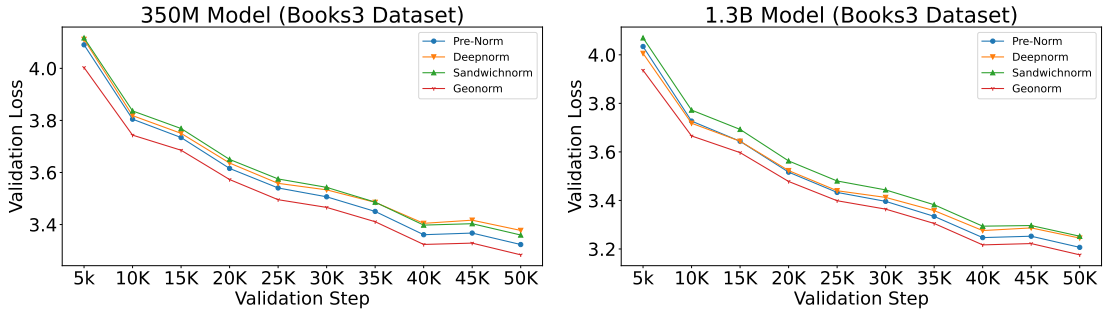

Figure 2: The performance of different baselines on the Books3 dataset, with model parameter 350M and 1.3B, training lengths of 512.

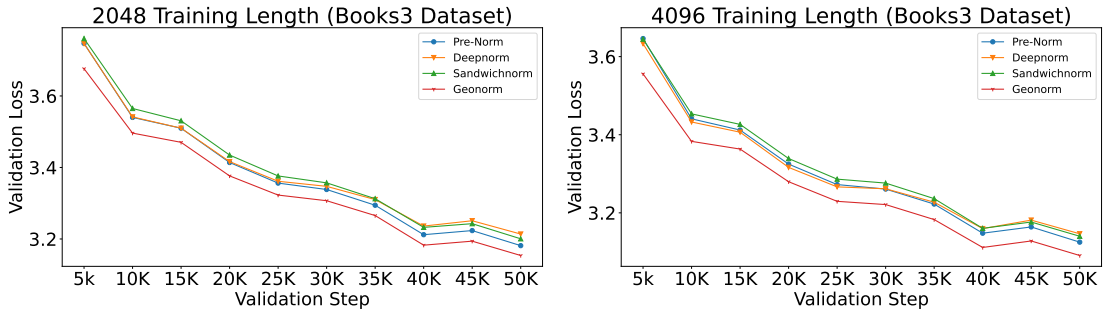

Figure 3: The performance of different baselines on the Books3 dataset, with model size 125M and training length 2048 and 4096.

both robust and generalizes effectively across key experimental parameters.

4.2 Performance on Large Model

The GeoNorm achieves the best performance with larger model size. As shown in Figure 2, the performance of GeoNorm demonstrates consistent superiority across increasing model sizes. For a 350M parameter model, it achieves the best loss of 3.2837, compared to 3.3233 (Pre-Norm), 3.3775 (DeepNorm), and 3.3598 (SandwichNorm). Scaling to 1.3B parameters, GeoNorm again leads with a loss of 3.1759, while the next best baseline, Pre-Norm, achieves only 3.2067. These results confirm two key points: first, GeoNorm delivers the best performance at every tested scale, and second, its performance advantage persists and even improves relative to alternatives as model size grows, showcasing its scalability.

4.3 Performance on Long Training Length

The GeoNorm keeps the best performance with longer training length. As shown in Figure 3, our results demonstrate the consistent superiority of the GeoNorm method across different training lengths. At a length of 2048, it achieves the best performance with a loss of 3.1538, outperforming Pre-Norm, DeepNorm, and SandwichNorm. Extending the training length to 4096 further improves its performance to a loss of 3.0908, while maintaining its lead over all baseline methods. This dual finding—performance improvement with extended training and sustained superiority—confirms that GeoNorm not only provides the best initial results but also retains its optimal performance advantage as training progresses.

4.4 The Effect of decay method

The GeoNorm with Hormonic Decay achieves better performance. As shown in Figure 4, the proposed GeoNorm method demonstrates significantly enhanced performance when utilizing the harmonic decay scheduling mechanism. Across multiple benchmark datasets, this configuration consistently yields state-of-the-art results. Specifically, on the Arxiv dataset, GeoNorm with harmonic decay achieves a loss of 1.8792, establishing a new performance benchmark. This represents a substantial improvement over alternative configurations, with the second-best performing method attaining only 1.8817. The performance improvement is even more pronounced on the larger and more complex Books3 dataset, where GeoNorm with harmonic decay records a loss of 3.4048,

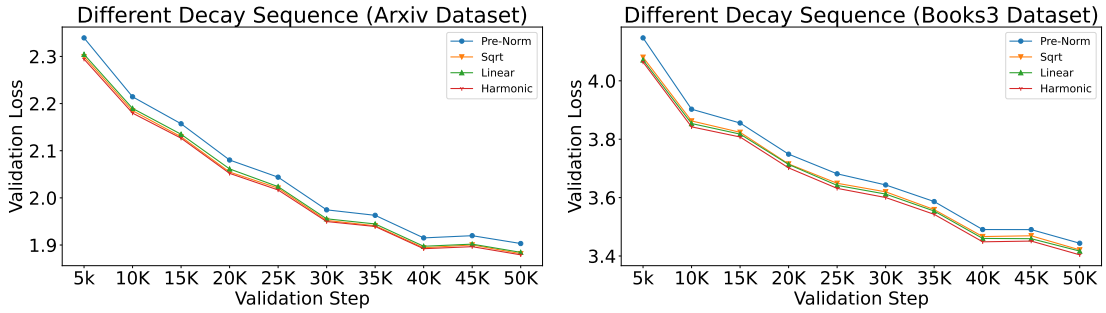

Figure 4: The performance of different decay methods on the Arxiv and Books3 dataset, with model parameter 125M, training lengths of 512. **Sqrt**: $\frac{\alpha}{k^{0.5}}$, **Linear**: $\frac{\alpha(T-k)}{T}$, and **Harmonic**: $\frac{\alpha}{k}$, where k is the current layer index and T is the total layer number.

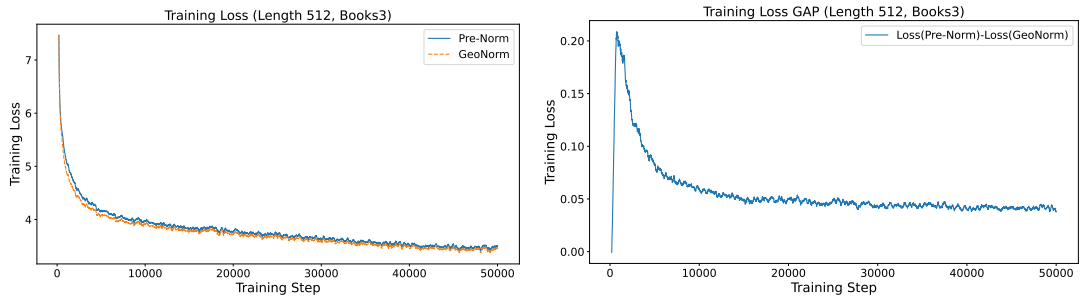

Figure 5: The training dynamic with loss metrics, with model parameter of 125M, training lengths of 512 and the Books3 dataset.

again outperforming all competing approaches. These empirical results collectively demonstrate that integrating harmonic decay scheduling with the GeoNorm imprroves the performance.

4.5 The Analysis of Training Dynamics

The GeoNorm achieves better performance across the training steps. As shown in Figure 5, the GeoNorm method establishes a decisive performance advantage early in the training phase, as evidenced by its significantly lower loss metrics. This initial lead is not merely temporary; GeoNorm demonstrates remarkable stability by maintaining its optimal performance throughout the entire training cycle without degradation. In direct comparison, the Pre-Norm baseline is consistently outperformed, confirming that GeoNorm's superiority is a persistent and defining characteristic of the training process from initialization to convergence.

The GeoNorm has faster loss reduction. The training dynamics reveal a significant loss gap with a peak value of 0.2. Crucially, this gap does not persist due to instability but consistently narrows as training progresses, converging to a much smaller final value of approximately 0.05 by the end of the run. This relatively large divergence indicates that the GeoNorm method successfully reduces its loss at a markedly faster rate during the training phase compared to the baseline, suggesting more efficient optimization.

4.6 The Performance on Large-Scale Pretrain Dataset

Downstream Evaluation. We evaluate performance on standard benchmarks, including ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), WinoGrade (Sakaguchi et al., 2021), SocialIQA (Sap et al., 2019) and BoolQ (Clark et al., 2019) using the `lm-evaluation-harness` (Gao et al., 2024) codebase. The evaluation metric is the accuracy. We train the model with 50K steps with training length 1024 and training tokens 50B. The model sizes are 350M, 760M, and 1.3B. The training length is 1024 and the training step is 50000. We display the zero-shot evaluation results of models here in Tables 1.

With the same data size, the GeoNorm is always better from small model size to a larger model size. With a fixed data size, GeoNorm reliably delivers the highest performance across all

Table 1: Main language modeling results against different methods. All models are trained on the same subset of the FineWeb-Edu dataset (Lozhkov et al., 2024) with the GPT-2 tokenizer.

Model	ARC-E	Hellaswag	OBQA	ScIQ	PIQA	Wino	SocialIQA	BoolQ	Avg
350M params/50B tokens									
Pre-Norm	44.87	44.85	28.60	68.70	69.80	52.80	**33.57**	59.11	50.29
Post-Norm	44.65	44.60	**30.80**	68.20	70.29	51.85	32.91	56.85	50.02
DeepNorm	42.80	39.31	29.60	66.00	67.79	49.96	32.40	60.21	48.51
SandwichNorm	43.98	43.40	30.40	67.30	71.11	52.09	32.91	60.18	50.17
GeoNorm	**44.99**	**45.79**	29.60	**69.20**	**71.27**	**53.20**	32.55	**60.83**	**50.93**
350M params/10B tokens									
Pre-Norm	43.52	40.23	**31.00**	64.80	68.61	49.64	32.80	55.14	48.22
Post-Norm	43.52	39.78	30.80	64.40	67.46	51.07	32.86	57.34	48.40
DeepNorm	42.13	36.71	28.40	64.90	66.81	49.41	33.52	**58.84**	47.59
SandwichNorm	42.97	39.55	30.00	65.90	**69.15**	51.78	**33.98**	55.84	48.65
GeoNorm	**44.15**	**40.99**	29.80	**67.40**	68.72	**51.85**	32.75	55.32	**48.87**
760M params/10B tokens									
Pre-Norm	46.42	45.47	31.20	69.00	69.75	52.41	**34.24**	56.61	50.64
Post-Norm	46.13	44.13	30.80	67.70	69.31	51.85	32.60	58.53	50.13
DeepNorm	44.02	41.07	30.80	68.00	68.82	51.93	33.42	60.00	49.76
SandwichNorm	**46.89**	44.75	31.40	69.20	69.59	51.30	33.11	58.47	50.59
GeoNorm	45.62	**46.14**	**31.80**	**69.40**	**71.27**	**52.64**	33.47	**60.61**	**51.37**
1.3B params/10B tokens									
Pre-Norm	47.35	48.29	30.60	70.90	70.89	**53.59**	33.27	**59.14**	51.75
Post-Norm	46.21	47.39	31.60	70.90	71.27	51.46	32.86	57.37	51.13
DeepNorm	45.88	44.14	31.60	68.90	70.62	53.12	33.21	58.90	50.80
SandwichNorm	46.63	47.61	30.60	71.10	70.73	52.57	**34.14**	58.17	51.44
GeoNorm	**49.28**	**49.30**	**31.80**	**73.50**	**71.33**	53.28	32.40	59.11	**52.50**

evaluated model sizes. With a 350M parameter model trained on 10B tokens, GeoNorm secures a leading average score of 48.87, marginally surpassing the second-best method, SandwichNorm (48.65). As model capacity escalates to 760M parameters—while keeping the training data constant at 10B tokens—the performance gap widens significantly. Here, GeoNorm achieves a superior average of 51.37, a clear lead over the 50.64 attained by Pre-Norm. The pattern of dominance culminates at the largest tested scale of 1.3B parameters, where GeoNorm again claims the top position with an average performance of 52.50, outperforming the second-best Pre-Norm (51.75) by an even more substantial margin. This unbroken sequence—from 350M to 1.3B parameters—confirms that GeoNorm's architectural advantage is intrinsic and scales effectively, consistently delivering better performance from small to large model sizes.

With the same model size, the GeoNorm achieves better performance from a small training data size to a large training data size. When trained on the relatively smaller dataset of 10B tokens, GeoNorm achieves an average of 48.87, which represents a 0.22-point advantage over SandwichNorm (48.65). When the training data for this identical architecture is increased fivefold to 50B tokens, the performance of all methods improves, yet GeoNorm's relative advantage grows. Its average score rises to 50.93, now establishing a more pronounced lead of 0.64 points over the new second-best performer, Pre-Norm (50.29). This result is critical: it demonstrates that GeoNorm possesses superior scaling laws with respect to training compute. Therefore, GeoNorm demonstrably achieves better performance from small to large training datasets for a given model size.

5 Conclusion

This work presents a unified view of Pre-Norm and Post-Norm through the lens of manifold optimization. Building on this formulation, we introduce GeoNorm, a novel approach that leverages geodesic optimization to improve model performance. We conduct a comprehensive evaluation of GeoNorm, testing it across multiple datasets, varying sequence lengths, different model sizes, and a range of downstream tasks. By advancing the theoretical understanding of normalization mechanisms, this research provides a foundation for subsequent innovations.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. *arXiv preprint arXiv:1808.07561*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. 2018. Understanding batch normalization. *Advances in neural information processing systems*, 31.

Léon Bottou. 1998. Online algorithms and stochastic approximations. *Online learning in neural networks*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yekun Chai, Shuo Jin, and Xinwen Hou. 2020. Highway transformer: Self-gating enhanced self-attentive networks. *arXiv preprint arXiv:2004.08178*.

François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Caio O da Silva, Yuri A Aoto, Felipe FGS Costa, and Márcio F da Silva. 2025. A thorough study of riemannian newton's method. *arXiv preprint arXiv:2506.09297*.

Peter Debye. 1909. Näherungsformeln für die zylinderfunktionen für große werte des arguments und unbeschränkt veränderliche werte des index. *Mathematische Annalen*, 67(4):535–558.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, and 1 others. 2023. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pages 7480–7512. PMLR.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and 1 others. 2021. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Yanwen Fang, Yuxi Cai, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. 2023. Cross-layer retrospective retrieving via layer attention. *arXiv preprint arXiv:2302.03985*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. 2020. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253.

Magnus R Hestenes, Eduard Stiefel, and 1 others. 1952. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436.

Jiang Hu, Xin Liu, Zai-Wen Wen, and Ya-Xiang Yuan. 2020. A brief introduction to manifold optimization. *Journal of the Operations Research Society of China*, 8(2):199–248.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 5254–5276.

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

Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. 2024. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. *arXiv preprint arXiv:2407.11239*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Akhil Kedia, Mohd Abbas Zaidi, Sushil Khyalia, Jungho Jung, Harshith Goka, and Haejun Lee. 2024. Transformers get stable: An end-to-end signal propagation theory for language models. *arXiv preprint arXiv:2403.09635*.

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

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Dimitrios Konstantinidis, Ilias Papastratis, Kosmas Dimitropoulos, and Petros Daras. 2023. Multi-manifold attention for vision transformers. *IEEE Access*, 11:123433–123444.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*.

Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. *arXiv preprint arXiv:2010.03737*.

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

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023a. Stack more layers differently: High-rank training through low-rank updates.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023b. Relora: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2024. Video-LLaVA: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5971–5984, Miami, Florida, USA. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020a. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. 2022a. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020b. Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022b. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. 2024. ngpt: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*.

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

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Brian Mak and Jeffrey Flanigan. 2025. Residual matrix transformers: Scaling the size of the residual stream. *arXiv preprint arXiv:2506.22696*.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204.

Stephen Menary, Samuel Kaski, and Andre Freitas. 2024. Transformer normalisation layers and the independence of semantic subspaces. *arXiv preprint arXiv:2406.17837*.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. 2024. Laurel: Learned augmented residual layer. *arXiv preprint arXiv:2411.07501*.

AI Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: 4-7-2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Toan Q Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. 2024. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. 2024. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Oleg Rybakov, Mike Chrzanowski, Peter Dykas, Jinze Xue, and Ben Lanir. 2024. Methods of improving llm training stability. *arXiv preprint arXiv:2410.16682*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Sam Shleifer, Jason Weston, and Myle Ott. 2021. Normformer: Improved transformer pretraining with extra normalization. *arXiv preprint arXiv:2110.09456*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. 2024. A deeper look at depth pruning of llms. *arXiv preprint arXiv:2407.16286*.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, and 1 others. 2025. A survey of reasoning with foundation models: Concepts, methodologies, and outlook. *ACM Computing Surveys*, 57(11):1–43.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023a. B2t connection: Serving stability and performance in deep transformers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3078–3095.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023b. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

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

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

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

Ting Xu, Zhichao Huang, Jiankai Sun, Shanbo Cheng, and Wai Lam. 2025. Seqpo-simt: Sequential policy optimization for simultaneous machine translation. *arXiv preprint arXiv:2505.20622*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yichun Yin, Wenyong Huang, Kaikai Song, Yehui Tang, Xueyu Wu, Wei Guo, Peng Guo, Yaoyuan Wang, Xiaojun Meng, Yasheng Wang, and 1 others. 2025. Pangu ultra: Pushing the limits of dense large language models on ascend npus. *arXiv preprint arXiv:2504.07866*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in neural information processing systems*, 32.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. *arXiv preprint arXiv:1908.11365*.

Hongyi Zhang and Suvrit Sra. 2016. First-order methods for geodesically convex optimization. In *Conference on learning theory*, pages 1617–1638. PMLR.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International conference on machine learning*, pages 11328–11339. PMLR.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. 2024. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Zhijian Zhuo, Yutao Zeng, Ya Wang, Sijun Zhang, Jian Yang, Xiaoqing Li, Xun Zhou, and Jinwen Ma. 2025. Hybridnorm: Towards stable and efficient transformer training via hybrid normalization. *arXiv preprint arXiv:2503.04598*.

A Limitations

This work mainly focuses on analyzing the Pre-Norm and Post-Norm, but it is not related to developing an application.

B AI Usage

This work uses the AI to polish the writing.

C Geodesic Optimization for Pre-Norm

Let $\boldsymbol{x}_k$ denote the token representation after the k-th Transformer layer. A Transformer layer with Pre-Norm can be expressed as:

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k + \phi\big(\mathrm{Norm}(\boldsymbol{x}_k)\big), \tag{6}$$

where $\phi(\cdot)$ denotes either the attention module $\mathrm{Attn}(\cdot)$ or the feed-forward network $\mathrm{FFN}(\cdot)$, and $\mathrm{Norm}(\cdot)$ represents a normalization operator (typically RMSNorm).

Let $\boldsymbol{x}_0 = \boldsymbol{x}$ be the initial input. After k layers of the Transformer model with a standard Pre-Norm, the representation can be written as

$$\boldsymbol{x}_k = \boldsymbol{x} + \phi_1(\mathrm{Norm}(\boldsymbol{x})) + \phi_2(\mathrm{Norm}(\boldsymbol{x}_1)) + \cdots + \phi_k(\mathrm{Norm}(\boldsymbol{x}_{k-1})). \tag{7}$$

This is obtained by recursively applying

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \phi_k\big(\mathrm{Norm}(\boldsymbol{x}_{k-1})\big), \tag{8}$$

for each layer index k. Here, for simplicity, we use $\phi_k(\cdot)$ to denote the attention and feed-forward module at the k-th layer. At the output of the final layer, an additional normalization operation is applied:

$$\text{FinalOutput} = \mathrm{Norm}(\boldsymbol{x}_k)\,. \tag{9}$$

We consider a widely used normalization method, namely **RMSNorm**, which is scale-invariant and normalizes vectors to have norm $\sqrt{D}$, where D denotes the hidden dimension. Specifically,

$$\mathrm{Norm}(\boldsymbol{v}) = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\sqrt{D}. \tag{10}$$

As a consequence, for any scalar $c > 0$,

$$\mathrm{Norm}(c\boldsymbol{v}) = \frac{c\boldsymbol{v}}{\|c\boldsymbol{v}\|}\sqrt{D} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\sqrt{D} = \mathrm{Norm}(\boldsymbol{v}), \tag{11}$$

showing that RMSNorm is invariant to positive rescaling. Hence, we have

$$\mathrm{Norm}\left(\frac{\boldsymbol{x}_k}{\sqrt{k+1}}\right) = \mathrm{Norm}(\boldsymbol{x}_k). \tag{12}$$

This implies that, under RMSNorm, dividing the representation by scalars such as $\sqrt{k+1}$ has **no effect** on the final normalized output.

In standard Pre-Norm architectures, the norm $\|\boldsymbol{x}_k\|$ typically grows with depth k due to the accumulation of residual updates in (8). While one might attempt to control this growth by rescaling $\boldsymbol{x}_k$ (e.g., dividing by $\sqrt{k+1}$) prior to normalization, such scaling is entirely eliminated by RMSNorm.

Complete Derivation of Alternative Scaled Update From the recursive update in (8), we observe that

$$\frac{\boldsymbol{x}_k}{\sqrt{k+1}} = \frac{\sqrt{k}}{\sqrt{k+1}}\frac{\boldsymbol{x}_{k-1}}{\sqrt{k}} + \frac{1}{\sqrt{k+1}}\phi_k\left(\mathrm{Norm}\left(\frac{\boldsymbol{x}_{k-1}}{\sqrt{k}}\right)\right).$$

Denote $\boldsymbol{x}_k^{\text{alt}}$ as $\frac{\boldsymbol{x}_k}{\sqrt{k+1}}$, we obtain the alternative scaled update rule

$$\boldsymbol{x}_k^{\text{alt}} = \frac{\sqrt{k}}{\sqrt{k+1}}\cdot\boldsymbol{x}_{k-1}^{\text{alt}} + \frac{1}{\sqrt{k+1}}\cdot\phi_k\big(\mathrm{Norm}(\boldsymbol{x}_{k-1}^{\text{alt}})\big)\,, \quad k \geq 1 \tag{13}$$

with initialization $\boldsymbol{x}_0^{\text{alt}} = \boldsymbol{x}_0$.

Equivalence After RMSNorm The final output of the Transformer model with RMSNorm is given by:

$$\text{FinalOutput} = \mathrm{RMSNorm}(\boldsymbol{x}_k) = \mathrm{RMSNorm}\big(\boldsymbol{x}_k^{\text{alt}}\big)\,, \tag{14}$$

which follows directly from the scale invariance of RMSNorm. Consequently, instead of analyzing the original dynamics $\boldsymbol{x}_k$, we may equivalently consider the rescaled sequence $\boldsymbol{x}_k^{\text{alt}}$, which admits the same final output up to a scalar normalization.

The update scheme in (13) shares the formulation with (3), with inadaptive angles $\theta = \arccos\left(\frac{\sqrt{k}}{\sqrt{k+1}}\right)$. In contrast, the proposed GeoNorm allows the adaptive angle, depending on the step size α_k, the current state $\boldsymbol{x}_k$, and the update direction $\boldsymbol{s}_k$. This is convincingly more practical and reliable in manifold optimization. We further verify the Pre-Norm and our GeoNorm interpretation and their output equivalence by conducting experiments on comparing their outputs. The following table shows the similar performance of Transformer models using Pre-Norm and Equation (13), which further supports the equivalence of these two formulations.

Taken together, the theoretical analysis and experimental evidence suggest that Pre-Norm can be interpreted as a special case of GeoNorm with a fixed, depth-dependent angle. This perspective

Table 2: The performance of different formulations of Pre-Norm update, with training length 512, Books3 dataset and loss metric.

Method	Seed	5K	10K	15K	20K	25K	30K	35K	40K	45K	50K
$x_k + \phi(Norm(x_k))$	1234	4.1470	3.9028	3.8555	3.7488	3.6815	3.6434	3.5865	3.4906	3.4904	3.4437
$\frac{\sqrt{k}}{\sqrt{k+1}} \cdot \boldsymbol{x}_k + \frac{1}{\sqrt{k+1}} \cdot \phi(Norm(\boldsymbol{x}_k))$	1234	4.1443	3.9012	3.8533	3.7491	3.6795	3.6454	3.5868	3.4894	3.4898	3.4430
$x_k + \phi(Norm(x_k))$	1235	4.1618	3.9471	3.8361	3.7779	3.6850	3.6297	3.5751	3.5354	3.4751	3.4548
$\frac{\sqrt{k}}{\sqrt{k+1}} \cdot \boldsymbol{x}_k + \frac{1}{\sqrt{k+1}} \cdot \phi(Norm(\boldsymbol{x}_k))$	1235	4.1643	3.9500	3.8377	3.7784	3.6885	3.6307	3.5752	3.5347	3.4738	3.4543
$x_k + \phi(Norm(x_k))$	1236	4.1537	3.9082	3.8357	3.7429	3.6990	3.6268	3.5140	3.4809	3.4542	3.4364
$\frac{\sqrt{k}}{\sqrt{k+1}} \cdot \boldsymbol{x}_k + \frac{1}{\sqrt{k+1}} \cdot \phi(Norm(\boldsymbol{x}_k))$	1236	4.1540	3.9070	3.8378	3.7417	3.7002	3.6286	3.5152	3.4816	3.4572	3.4381
$x_k + \phi(Norm(x_k))$	mean	4.1542	3.9194	3.8424	3.7565	3.6885	3.6333	3.5585	3.5023	3.4732	3.4450
	std	0.0061	0.0197	0.0092	0.0153	0.0076	0.0072	0.0318	0.0237	0.0148	0.0076
$\frac{\sqrt{k}}{\sqrt{k+1}} \cdot \boldsymbol{x}_k + \frac{1}{\sqrt{k+1}} \cdot \phi(Norm(\boldsymbol{x}_k))$	mean	4.1542	3.9194	3.8429	3.7564	3.6894	3.6349	3.5591	3.5019	3.4736	3.4451
	std	0.0082	0.0218	0.0073	0.0158	0.0085	0.0075	0.0314	0.0234	0.0133	0.0068

Table 3: The performance of different clamp values for the implementation in Appendix F

Model	5K	10K	15K	20K	25K	30K	35K	40K	45K	50K
$\pi/2$	4.0650	3.8464	3.8096	3.7072	3.6341	3.6060	3.5471	3.4537	3.4531	3.4092
$\pi/4$	4.0642	**3.8423**	**3.8076**	**3.7018**	**3.6317**	**3.6003**	**3.5427**	**3.4486**	**3.4511**	**3.4040**
$\pi/8$	**4.0641**	3.8501	3.8145	3.7140	3.6471	3.6141	3.5544	3.4638	3.4631	3.4182

Table 4: **Model Configurations.**

	125M	350M
Training sequence length	512	512
Batch size	32	32
Number of iterations	50k	50k
Dropout prob.	0.0	0.0
Attention dropout prob.	0.0	0.0
Attention head	12	16
Feature dimension	768	1024
Layer number	12	24
Optimizer	Adam	Adam
Optimizer parameter betas	[0.9, 0.95]	[0.9, 0.95]
Learning rate	$6e-4$	$3e-4$
Precision	float16	float16

also sheds light on the empirical stability advantages of Pre-Norm over Post-Norm. In particular, Pre-Norm implicitly introduces a form of step size decay through its normalized residual structure, whereas Post-Norm lacks such a mechanism. From an optimization perspective, this absence may lead to instability or divergence when step sizes are not carefully controlled, explaining the commonly observed loss spikes and training collapse in Post-Norm Transformers.

D The Effect of Clamp Value

We have discussed the effect of the clamp value in Table 3. With clamp value $\pi/4$, the model achieves the best performance.

E Model Configuration

All experiments are conducted on 8 GPUs. The 125M and 350M model configuration is the following.

F Implementation Details

In this section, we present the implementation of the proposed GeoNorm module in PyTorch.

```
from tqdm import tqdm
import numpy as np

import torch
import torch.nn as nn

class GeoNorm(nn.Module):
    def __init__(self,clamp=torch.pi/4, method_name="harnomic"):
        super().__init__()

        self.scale=nn.Parameter(torch.tensor(1))
        self.bias=nn.Parameter(torch.tensor(0))
        self.clamp=torch.pi/4
        self.method_name=method_name
    def forward(self,x,g,layer_number,layer_total):
        """
        x: residual;
        g: ffn(x) or attn(x);
        layer_number: the current layer index, from 0,1,2, 3 to layer_total-1
        layer_total: layer total number.
        """
        gradient = g - (x * g).sum(dim=-1,keepdim=True) / (
                                            torch.norm(x, p=2, dim=-1, keepdim
                                                =True) ** 2) * x


        tangent_norm = torch.norm(gradient, p=2, dim=-1, keepdim=True)
        safe_tangent_norm = torch.clamp(tangent_norm, min=1e-8)
        unit_tangent = gradient / safe_tangent_norm

        R = torch.norm(x, p=2, dim=-1, keepdim=True)
        safe_R = torch.clamp(R, min=1e-6)
        theta = torch.clamp(safe_tangent_norm / safe_R, max=self.clamp)
        if self.method_name=="harmonic":
            theta_attn = torch.clamp((theta* self.scale +self.bias) / (
                layer_number + 1) , max=self.clamp)
        elif self.method_name=="linear":
            theta = torch.clamp((theta* self.scale +self.bias)*(layer_total-
                layer_number) / (layer_total) , max=self.clamp)
        elif self.method_name=="sqrt":
            theta = torch.clamp((theta* self.scale +self.bias) / math.sqrt(
                layer_number + 1) , max=self.clamp)

        output = x * torch.cos(theta) + unit_tangent * safe_R * torch.sin(
            theta)

        return output
```