THE CURSE OF DEPTH IN LARGE LANGUAGE MODELS

Wenfang Sun [1]*, **Xinyuan Song** [2]*, **Pengxiang Li**[3]*, **Lu Yin**[4], **Yefeng Zheng**[1], **Shiwei Liu**[5]†
[1] Westlake University [2] Emory University [3] Dalian University of Technology
[4] University of Surrey [5] University of Oxford

ABSTRACT

In this paper, we introduce *the Curse of Depth*, a concept that highlights, explains, and addresses the recent observation in modern Large Language Models (LLMs) where nearly half of the layers are less effective than expected. We first confirm the wide existence of this phenomenon across the most popular families of LLMs such as Llama, Mistral, DeepSeek, and Qwen. Our analysis, theoretically and empirically, identifies that the underlying reason for the ineffectiveness of deep layers in LLMs is the widespread usage of Pre-Layer Normalization (Pre-LN). While Pre-LN stabilizes the training of Transformer LLMs, its output variance exponentially grows with the model depth, which undesirably causes the derivative of the deep Transformer blocks to be an identity matrix, and therefore barely contributes to the training. To resolve this training pitfall, we propose LayerNorm Scaling, which scales the variance of output of the layer normalization inversely by the square root of its depth. This simple modification mitigates the output variance explosion of deeper Transformer layers, improving their contribution. Our experimental results, spanning model sizes from 130M to 1B, demonstrate that LayerNorm Scaling significantly enhances LLM pre-training performance compared to Pre-LN. Moreover, this improvement seamlessly carries over to supervised fine-tuning. All these gains can be attributed to the fact that LayerNorm Scaling enables deeper layers to contribute more effectively during training.

1 INTRODUCTION

Recent studies reveal that the deeper layers (Transformer blocks) in modern LLMs tend to be less effective than the earlier ones (Yin et al., 2023; Gromov et al., 2024; Men et al., 2024). On the one hand, this interesting observation provides an effective indicator for LLM compression. For instance, we can compress deeper layers significantly more (Yin et al., 2023; Lu et al., 2024; Dumitru et al., 2024) to achieve high compression ratios. Even more aggressively, entire deep layers can be pruned completely without compromising performance for the sake of more affordable LLMs (Muralidharan et al., 2024; Siddiqui et al., 2024). On the other hand, having many layers ineffective is undesirable as modern LLMs are extremely resource-intensive to train, often requiring thousands of GPUs trained for multiple months, let alone the labor used for data curation and administration Achiam et al. (2023); Touvron et al. (2023). Ideally, we want all layers in a model to be well-trained, with sufficient diversity in features from layer to layer, to maximize the utility of resources (Li et al., 2024b). The existence of ill-trained layers suggests that there must be something off with current LLM paradigms. Addressing such limitations is a pressing need for the community to avoid the waste of valuable resources, as new versions of LLMs are usually trained with their previous computing paradigm which results in ineffective layers. To seek the immediate attention of the community, we introduce the concept of *the Curse of Depth (CoD)* to systematically present the phenomenon of ineffective deep layers in various LLM families, to identify the underlying reason behind it, and to rectify it by proposing LayerNorm Scaling. We first state *the Curse of Depth* below.

The Curse of Depth. *The Curse of Depth* refers to the observed phenomenon where deeper layers in modern large language models (LLMs) contribute significantly less to learning and representation compared to earlier layers. These deeper layers often exhibit remarkable robustness to pruning and perturbations, implying they fail to perform meaningful transformations. This behavior pre-

*Equal contribution. †Corresponding to Shiwei Liu, shiwei.liu@maths.ox.ac.uk.

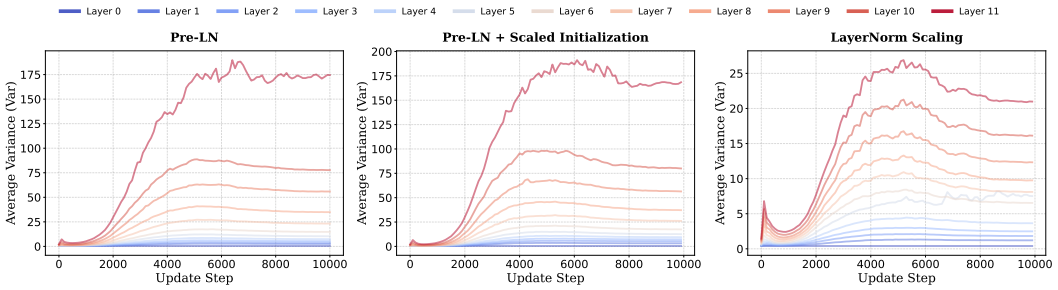

Figure 1: **Layerwise output variance.** This figure compares the output variance across various layers for different setups: (1) Pre-LN; (2) Pre-LN with Scaled Initialization; and (3) LayerNorm Scaling. The experiments are conducted on the LLaM-130M model trained for 10,000 steps.

vents these layers from effectively contributing to training and representation learning, resulting in resource inefficiency.

Empirical Evidence of CoD. The ineffectiveness of deep layers in LLMs has been previously reported. Yin et al. (2023) found that deeper layers of LLMs can tolerate significantly higher levels of pruning compared to shallower layers, achieving high sparsity. Similarly, Gromov et al. (2024) and Men et al. (2024) demonstrated that removing early layers causes a dramatic decline in model performance, whereas removing deep layers does not. Lad et al. (2024) showed that the middle and deep layers of GPT-2 and Pythia exhibit remarkable robustness to perturbations such as layer swapping and layer dropping. Recently, Li et al. (2024a) highlighted that early layers contain more outliers and are therefore more critical for fine-tuning. While these studies effectively highlight the limitations of deep layers in LLMs, they stop short of identifying the root cause of this issue or proposing viable solutions to address it. To demonstrate that *the Curse of Depths* is prevalent across popular families of LLMs, we conduct layer pruning experiments on various models, including LLaMA2-7/13B, Mistral-7B, DeepSeek-7B, and Qwen-7B. We measure performance degradation on the Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2020) by pruning entire layers of each model, one at a time, and directly evaluating the resulting pruned models on MMLU without any fine-tuning in Figure 2. *Results:* 1). Most LLMs utilizing Pre-LN exhibit remarkable robustness to the removal of deeper layers, whereas BERT with Post-LN shows the opposite trend. 2). The number of layers that can be pruned without significant performance degradation increases with model size.

Identifying the Root Cause of CoD. We theoretically and empirically identify the root cause of CoD as the use of Pre-Layer Normalization (Pre-LN) (Baevski & Auli, 2018; Dai, 2019), which normalizes layer inputs before applying the main computations, such as attention or feedforward operations, rather than after. Specifically, while stabilizing training, we observe that the output variance of Pre-LN accumulates significantly with layer depth (see Appendix D), causing the derivatives of deep Pre-LN layers to approach an identity matrix. This behavior prevents these layers from introducing meaningful transformations, leading to diminished representation learning.

Mitigating CoD through LayerNorm Scaling. We propose LayerNorm Scaling, which scales the output of Layer Normalization by the square root of the depth $\frac{1}{\sqrt{l}}$. LayerNorm Scaling effectively scales down the output variance across layers of Pre-LN, leading to considerably lower training loss and achieving the same loss as Pre-LN using only half tokens. Figure 1 compares the layerwise output variance across different setups: (1) Pre-LN, (2) Pre-LN with Scaled Initialization (Takase et al., 2023), and (3) LayerNorm Scaling. As shown, Pre-LN exhibits significant variance explosion in deeper layers. In contrast, LayerNorm Scaling effectively reduces output variance across layers, enhancing the contribution of deeper layers during training. This adjustment leads to significantly lower training loss compared to Pre-LN. Unlike previous LayerNorm variants (Li et al., 2024b; Liu et al., 2020), LayerNorm Scaling is simple to implement, requires no hyperparameter tuning, and introduces no additional parameters during training. Furthermore, we further show that the model pre-trained with LayerNorm Scaling achieves better performance on downstream tasks in self-supervised fine-tuning, all thanks to the more effective deep layers learned.

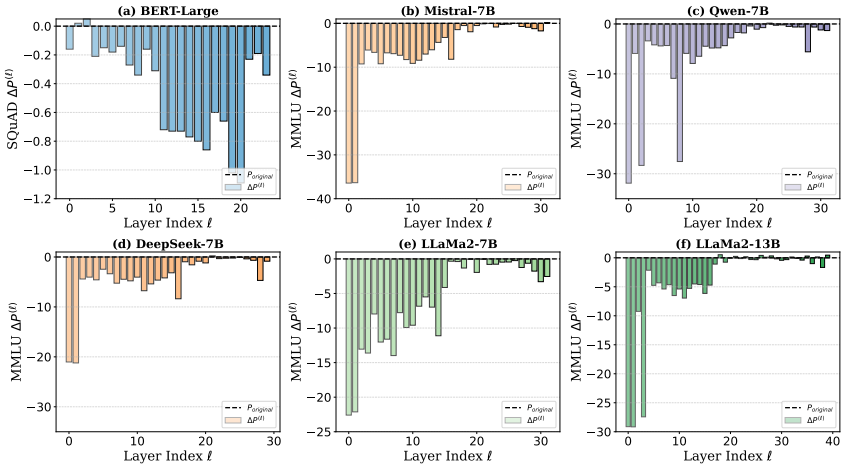

Figure 2: **Performance drop of layer pruning across different LLMs.**

2 Empirical Evidence of the Curse of Depth

To analyze how layer normalization affects the *Curse of Depth*, we follow and extend the setup of (Li et al., 2024b) to compare Pre-LN and Post-LN models.

Methods: We evaluate Pre-LN and Post-LN models by assessing the impact of layer pruning at different depths. Our hypothesis is that Pre-LN models exhibit diminishing effectiveness in deeper layers, whereas Post-LN has less effective early layers. To verify this, we empirically quantify the contribution of individual layers to overall model performance across a diverse set of LLMs, including BERT-Large (Devlin, 2018), Mistral-7B (Jiang et al., 2023), LLaMA2-7B/13B (Touvron et al., 2023), DeepSeek-7B (Bi et al., 2024), and Qwen-7B (Bai et al., 2023). These models were chosen to ensure architectural and application diversity. BERT-Large represents a Post-LN model, whereas the rest are Pre-LN-based. BERT-Large, a Post-LN model, is evaluated on SQuAD v1.1 (Rajpurkar, 2016), while others are assessed using the MMLU benchmark (Hendrycks et al., 2020).

Figure 2 presents the performance drop ($\Delta P^{(\ell)}$) across different layers for six LLMs, including one Post-LN model (BERT-Large) and five Pre-LN models (Mistral-7B, LLaMA2-13B, Qwen-7B, DeepSeek-7B and LLaMA2-7B). As shown in Figure 2 (a), pruning deeper layers in BERT-Large leads to a significant decline in accuracy on SQuAD v1.1, while pruning earlier layers has minimal impact. The performance drop $\Delta P^{(\ell)}$ becomes particularly severe beyond the 10th layer, highlighting the crucial role of deeper layers in maintaining overall performance in Post-LN models. In contrast, removing layers in the first half of the network results in negligible changes, indicating their limited contribution to the final output. However, as shown in Figure 2 (b)-(f), Pre-LN models exhibit a contrast pattern, where deeper layers contribute significantly less to the overall model performance. For instance, as shown in Figure 2 (b) and (c), pruning layers in the last third of Mistral-7B and Qwen-7B results in a minimal performance drop on MMLU, indicating their limited contribution to overall accuracy. In contrast, pruning the first few layers leads to a substantial accuracy degradation, highlighting their crucial role in feature extraction. Similarly, Figure 2 (d) and (e) show that DeepSeek-7B and LLaMA2-7B follow a similar pattern, where deeper layers have little impact on performance, while earlier layers play a more significant role. Finally, as shown in Figure 2 (f), more than half of the layers in LLaMA2-13B can be safely removed. This observation underscores the need for the community to address the *Curse of Depth* to prevent resource waste.

3 LayerNorm Scaling

Our theoretical(see Appendix A) and empirical analyses indicate that Pre-LN amplifies output variance, leading to the *Curse of Depth* and reducing the effectiveness of deeper layers. To mitigate this issue, we propose LayerNorm Scaling, a simple yet effective normalization strategy. The core idea of LayerNorm Scaling is to control the exponential growth of output variance in Pre-LN by scaling the normalized outputs according to layer depth. Specifically, we apply a scaling factor inversely proportional to the square root of the layer index to scale down the output of LN layers, stabilizing gradient flow and enhancing the contribution of deeper Transformer layers during

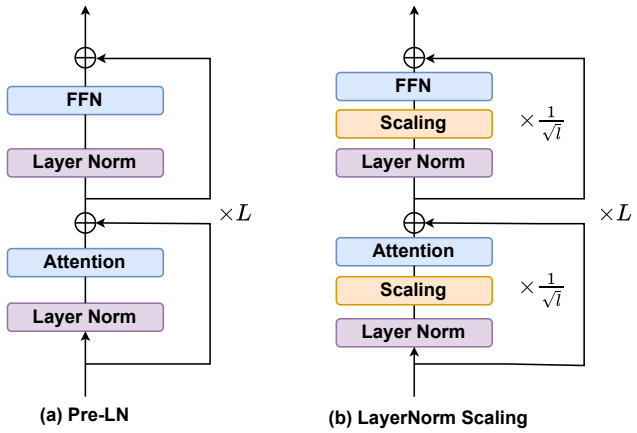

Figure 3: **Comparison between Pre-LN (a) and LayerNorm Scaling (b).**

training. LayerNorm Scaling is illustrated in Figure 3. Formally, for a Transformer model with L layers, the output of Layer Normalization in each layer ℓ is scaled by a factor of $\frac{1}{\sqrt{\ell}}$. Let $\mathbf{h}^{(\ell)}$ denote the input to Layer Normalization at layer ℓ. The modified output is computed as:

$$\mathbf{h}^{(\ell)} = \text{LayerNorm}(\mathbf{h}^{(\ell)}) \times \frac{1}{\sqrt{\ell}}, \tag{1}$$

where $\ell \in \{1, 2, \ldots, L\}$. This scaling prevents excessive variance growth with depth, addressing a key limitation of Pre-LN. Unlike Mix-LN, which stabilizes gradients in deeper layers but suffers from training instability caused by Post-LN (Nguyen & Salazar, 2019; Wang et al., 2024), LayerNorm Scaling preserves the stability advantages of Pre-LN while enhancing the contribution of deeper layers to representation learning. Applying LayerNorm Scaling leads to a notable reduction of layerwise output variance, resulting in lower training loss and faster convergence than Pre-LN.

4 Experiments

4.1 LLM Pre-training

Table 1: Perplexity (↓) comparison of layer normalization methods across LLaMA sizes.

	LLaMA-130M	**LLaMA-250M**	**LLaMA-350M**	**LLaMA-1B**
Training Tokens	2.2B	3.9B	6.0B	8.9B
Post-LN (Ba, 2016)	26.95	1409.79	1368.33	1390.75
DeepNorm (Wang et al., 2024)	27.17	22.77	1362.59	1409.08
Mix-LN (Li et al., 2024b)	26.07	21.39	1363.21	1414.78
Pre-LN (Baevski & Auli, 2018)	26.73	21.92	19.58	17.02
Pre-LN + LayerNorm Scaling	**25.76**	**20.35**	**18.20**	**15.71**

we follow the setup of Li et al. (2024b), using the same model configurations and training conditions to compare it with normalization methods like Post-LN (Nguyen & Salazar, 2019), DeepNorm (Wang et al., 2024), and Pre-LN (Dai, 2019). Experiments are conducted on LLaMA-based models (130M, 250M, 350M, and 1B parameters) with consistent architecture and training settings, as in Lialin et al. (2023) and Zhao et al. (2024). All models use RMSNorm (Shazeer, 2020) and SwiGLU (Zhang & Sennrich, 2019) activations, with Adam optimizer (Kingma, 2014). All models share the same architecture, hyperparameters, and training schedule, with the only difference being the choice of normalization method. Unlike Mix-LN (Li et al., 2024b), which introduces an additional hyperparameter α manually set to 0.25, LayerNorm Scaling requires no extra hyperparameters, making it simpler to implement. Table 1 shows that LayerNorm Scaling consistently outperforms other normalization methods across different model sizes. While DeepNorm performs comparably to Pre-LN on smaller models, it struggles with larger architectures like LLaMA-1B, showing signs of instability and divergence in loss values. Similarly, Mix-LN outperforms Pre-LN in smaller models but faces convergence issues with LLaMA-350M, indicating its sensitivity to

Table 2: Fine-tuning performance (↑) of LLaMA with various normalizations.

Method	MMLU	BoolQ	ARC-e	PIQA	Hellaswag	OBQA	Winogrande	Average
LLaMA-250M								
Post-LN (Ba, 2016)	22.95	37.83	26.94	52.72	26.17	11.60	49.56	32.54
DeepNorm (Wang et al., 2024)	23.60	37.86	36.62	61.10	25.69	15.00	49.57	35.63
Mix-LN (Li et al., 2024b)	26.53	56.12	41.68	66.34	30.16	18.00	50.56	41.34
Pre-LN (Baevski & Auli, 2018)	24.93	38.35	40.15	63.55	26.34	16.20	49.01	36.93
Pre-LN + LayerNorm Scaling	**27.08**	**58.17**	**45.24**	**67.38**	**32.81**	**18.80**	**52.49**	**43.14**
LLaMA-1B								
Post-LN (Ba, 2016)	22.95	37.82	25.08	49.51	25.04	13.80	49.57	31.96
DeepNorm (Wang et al., 2024)	23.35	37.83	27.06	52.94	26.19	11.80	49.49	32.67
Mix-LN (Li et al., 2024b)	23.19	37.83	25.08	49.51	25.04	11.80	49.57	31.72
Pre-LN (Baevski & Auli, 2018)	26.54	**62.20**	45.70	67.79	30.96	17.40	50.51	43.01
Pre-LN + LayerNorm Scaling	**28.69**	61.80	**48.85**	**67.92**	**33.94**	**18.60**	**54.30**	**44.87**

architecture design and hyperparameter tuning due to the introduction of Post-LN. Notably, Mix-LN was originally evaluated on LLaMA-1B with 50,000 steps (Li et al., 2024b), while our setting extends training to 100,000 steps, where Mix-LN fails to converge, highlighting its instability in large-scale settings caused by the usage of Post-LN.

In contrast, LayerNorm Scaling solves the *Curse of Depth* without compromising the training stability thanks to its simplicity. LayerNorm Scaling achieves the lowest perplexity across all tested model sizes, showing stable performance improvements over existing methods. For instance, on LLaMA-130M and LLaMA-1B, LayerNorm Scaling reduces perplexity by 0.97 and 1.31, respectively, compared to Pre-LN. Notably, LayerNorm Scaling maintains stable training dynamics for LLaMA-1B, a model size where Mix-LN fails to converge. These findings demonstrate that LayerNorm Scaling provides a robust and computationally efficient normalization strategy, enhancing large-scale language model training without additional implementation complexity.

4.2 Supervised Fine-tuning

We believe that LayerNorm Scaling allows deeper layers in LLMs to contribute more effectively during supervised fine-tuning by alleviating gradient vanishing associated with increasing depth. Compared to models trained with Pre-LN, the deeper layers with LayerNorm Scaling maintain stable output variance, preventing uncontrolled growth and ensuring effective feature representation. As a result, deeper layers contribute more effectively to feature transformation, enhancing representation learning and improving generalization on complex downstream tasks.

To verify this, we follow the fine-tuning methodologies in Li et al. (2024b) and Li et al. (2024a), applying the same optimization settings as pre-training. We fine-tune models from Section 4.1 on the Commonsense170K dataset (Hu et al., Hu et al. (2023)) across eight downstream tasks. The results, presented in Table 2, demonstrate that LayerNorm Scaling consistently surpasses other normalization techniques in all evaluated datasets. For the LLaMA-250M model, LayerNorm Scaling improves average performance by1.80% and achieves a 3.56% gain on ARC-e compared to Mix-LN. Similar trends are observed with the LLaMA-1B model, where LayerNorm Scaling outperforms Pre-LN, Post-LN, Mix-LN, and DeepNorm on seven out of eight tasks, with an average gain of 1.86% over the best baseline. These results confirm that LayerNorm Scaling, by improving gradient flow and deep-layer representation quality, achieves better fine-tuning performance, demonstrating robustness and enhanced generalization on diverse downstream tasks.

5 Conclusion

In this paper, we introduce the concept of the *Curse of Depth* in LLMs, highlighting an urgent yet often overlooked phenomenon: nearly half of the deep layers in modern LLMs are less effective than expected. We discover the root cause of this phenomenon is Pre-LN which is widely used in almost all modern LLMs. To tackle this issue, we introduce *LayerNorm Scaling*. By scaling the output variance inversely with the layer depth, LayerNorm Scaling ensures that all layers, including deeper ones, contribute meaningfully to training. Our experiments show that this simple modification improves performance, reduces resource usage, and stabilizes training across various model sizes. LayerNorm Scaling is easy to implement, hyperparameter-free, and provides a robust solution to enhance the efficiency and effectiveness of LLMs.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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.

Zihang Dai. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Razvan-Gabriel Dumitru, Vikas Yadav, Rishabh Maheshwary, Paul-Ioan Clotan, Sathwik Tejaswi Madhusudhan, and Mihai Surdeanu. Layer-wise quantization: A pragmatic and effective method for quantizing llms beyond integer bit-levels. *arXiv preprint arXiv:2406.17415*, 2024.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

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.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vedang Lad, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm fine-tuning. *arXiv preprint arXiv:2405.18380*, 2024a.

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

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2023.

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

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *NeurIPS*, 2024.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Bhuminand Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *NeurIPS*, 2024.

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

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

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

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*, 2023.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

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*, 2024.

E. T. Whittaker and G. N. Watson. *A Course of Modern Analysis*. Cambridge Mathematical Library. Cambridge University Press, 4 edition, 1996.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

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

A THEORETICAL ANALYSIS OF CURSE OF DEPTH

A.1 PRELIMINARIES

This paper primarily focuses on Pre-LN Transformer (Baevski & Auli, 2018; Dai, 2019). Let $x_\ell \in \mathbb{R}^d$ be the input vector at the ℓ-th layer of Transformer, where d denotes the feature dimension of each layer. For simplicity, we assume all layers to have the same dimension d. The layer output y is calculated as follows:

$$y = x_{\ell+1} = x'_\ell + \mathrm{FFN}(\mathrm{LN}(x'_\ell)), \tag{2}$$

$$x'_\ell = x_\ell + \mathrm{Attn}(\mathrm{LN}(x_\ell)), \tag{3}$$

where LN denotes the layer normalization function. In addition, the feed-forward network (FFN) and the multi-head self-attention (Attn) sub-layers are defined as follows:

$$\begin{aligned} \mathrm{FFN}(x) &= W_2\mathcal{F}(W_1 x), \\ \mathrm{Attn}(x) &= W_O(\mathrm{concat}(\mathrm{head}_1(x), \ldots, \mathrm{head}_h(x))), \\ \mathrm{head}_i(x) &= \mathrm{softmax}\left(\frac{(W_{Qi}x)^\top (W_{Ki}X)}{\sqrt{d_{\mathrm{head}}}}\right)(W_{Vi}X)^\top, \end{aligned} \tag{4}$$

where $\mathcal{F}$ is an activation function, concat concatenates input vectors, $\mathrm{softmax}$ applies the softmax function, and $W_1 \in \mathbb{R}^{d_{\mathrm{ffn}} \times d}$, $W_2 \in \mathbb{R}^{d \times d_{\mathrm{ffn}}}$, $W_{Qi} \in \mathbb{R}^{d_{\mathrm{head}} \times d}$, $W_{Ki} \in \mathbb{R}^{d_{\mathrm{head}} \times d}$, $W_{Vi} \in \mathbb{R}^{d_{\mathrm{head}} \times d}$, and $W_O \in \mathbb{R}^{d \times d}$ are parameter matrices, and d_{FFN} and d_{head} are the internal dimensions of FFN and multi-head self-attention sub-layers, respectively. $X \in \mathbb{R}^{d \times s}$, where s is the input sequence length.

The derivatives of Pre-Ln Transformers are:

$$\frac{\partial \text{Pre-LN}(x)}{\partial x} = I + \frac{\partial f(\text{LN}(x))}{\partial \text{LN}(x)} \frac{\partial \text{LN}(x)}{\partial x}, \tag{5}$$

where f here represents either the multi-head attention function or the FFN function. If the term $\frac{\partial f(\text{LN}(x))}{\partial \text{LN}(x)} \frac{\partial \text{LN}(x)}{\partial x}$ becomes too small, the Pre-LN layer $\frac{\partial \text{Pre-LN}(x)}{\partial x}$ behaves like an identity map. Our main objective is to prevent identity map behavior for very deep Transformer networks. The first step in this process is to compute the variance $\sigma^2_{x_\ell}$ of vector x_ℓ.

A.2 PRE-LN TRANSFORMERS

Assumption 1. *Let x_ℓ and x'_ℓ denote the input and intermediate vectors of the ℓ-th layer. Moreover, let W_ℓ denote the model parameter matrix at the ℓ-th layer. We assume that, for all layers, x_ℓ, x'_ℓ, and W_ℓ follow normal and independent distributions with mean $\mu = 0$.*

Lemma 1. *Let $\sigma^2_{x'_\ell}$ and $\sigma^2_{x_\ell}$ denote the variances of x'_ℓ and x_ℓ, respectively. These two variances exhibit the same overall growth trend, which is:*

$$\sigma^2_{x_\ell} = \sigma^2_{x_1} \Theta\Big(\prod_{k=1}^{\ell-1}\left(1 + \frac{1}{\sigma_{x_k}}\right)\Big), \tag{6}$$

where the growth of $\sigma^2_{x_l}$ is sub-exponential, as shown by the following bounds:

$$\Theta(L) \leq \sigma^2_{x_l} \leq \Theta(\exp(L)). \tag{7}$$

Here, the notation Θ means: if $f(x) \in \Theta\big(g(x)\big)$, then there exist constants C_1, C_2 such that $C_1 |g(x)| \leq |f(x)| \leq C_2 |g(x)|$ as $x \to \infty$. The lower bound $\Theta(L) \leq \sigma^2_{x_\ell}$ indicates that $\sigma^2_{x_\ell}$ grows at least linearly, while the upper bound $\sigma^2_{x_\ell} \leq \Theta(\exp(L))$ implies that its growth does not exceed an exponential function of L.

Based on Assumption 1 and the work of Takase et al. (2023), we obtain the following:

Theorem 1. *For a Pre-LN Transformer with L layers, using equation 2 and equation 3, the partial derivative $\frac{\partial y_L}{\partial x_1}$ can be written as:*

$$\frac{\partial y_L}{\partial x_1} = \prod_{\ell=1}^{L-1} \left(\frac{\partial y_\ell}{\partial x'_\ell} \cdot \frac{\partial x'_\ell}{\partial x_\ell} \right). \tag{8}$$

The Euclidean norm of $\frac{\partial y_L}{\partial x_1}$ is given by:

$$\left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq \prod_{l=1}^{L-1} \left(1 + \frac{1}{\sigma_{x_\ell}} A + \frac{1}{\sigma_{x_\ell}^2} B \right), \tag{9}$$

where A and B are constants for the Transformer network. Then the upper bound for this norm is given as follows: when $\sigma_{x_\ell}^2$ grows exponentially, (i.e., at its upper bound), we have:

$$\sigma_{x_\ell}^2 \sim \exp(\ell), \quad \left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq M, \tag{10}$$

where the gradient norm converges to a constant M. Conversely, when $\sigma_{x_\ell}^2$ grows linearly (i.e., at its lower bound), we have

$$\sigma_{x_\ell}^2 \sim \ell, \quad \left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq \Theta(L), \tag{11}$$

which means that the gradient norm grows linearly in L.

The detailed description of A and B, as well as the complete proof, are provided in Appendix A.5.

From Theorem 1, we observe that when the variance grows exponentially, as the number of layers $L \to \infty$, the norm $\left\| \frac{\partial y_L}{\partial x_1} \right\|_2$ is bounded above by a fixed constant M. This result implies that even an infinitely deep Transformer remains stable, and by the Weierstrass Theorem, the network is guaranteed to converge. Consequently, this implies that for very large L, deeper layers behave nearly as an **identity map** from x_ℓ to y_ℓ, thereby limiting the model's expressivity and hindering its ability to learn meaningful transformations.

This outcome is undesirable, therefore, we would instead prefer the variance to increase more gradually—e.g., linearly—so that $\left\| \frac{\partial y_L}{\partial x_1} \right\|_2$ exhibits linear growth. This observation highlights the necessity of appropriate variance control mechanisms, such as scaling strategies, to prevent excessive identity mappings and enhance network depth utilization.

A.3 Post-LN Transformers

For Post-LN Transformers, we continue to adopt Assumption 1. In this setting, each layer is followed by a layer normalization (LN) step, ensuring that the variances $\sigma_{x_\ell}^2$ and $\sigma_{x'_\ell}^2$ remain fixed at 1 across all layers. Consequently, the norm $\left\| \frac{\partial y_\ell}{\partial x_\ell} \right\|_2$ exhibits minimal variation from one layer to the next, indicating stable gradient propagation.

Since the variance is effectively controlled by LN in Post-LN Transformers, an explicit variance-based analysis becomes less critical. Nonetheless, there remain other important aspects to investigate in deeper Post-LN architectures, such as the evolution of feature mappings and the behavior of covariance kernels over deep layers. These directions will be pursued in future work.

A.4 Proof of Lemma 1

Proof. Given equation 2 from Takase et al. (2023), we have:

$$\begin{aligned} y = x_{\ell+1} &= x'_\ell + \mathrm{FFN}(\mathrm{LN}(x'_\ell)), \\ x'_\ell &= x_\ell + \mathrm{Attn}(\mathrm{LN}(x_\ell)). \end{aligned} \tag{12}$$

Based on our Assumption 1, let $\mathrm{Var}(\mathrm{Attn}(\mathrm{LN}(x_\ell))) = \sigma^2_{\text{Attn}}$. Then we can write:

$$\begin{aligned}\mathrm{Var}(x'_\ell) &= \mathrm{Var}(x_\ell) + \mathrm{Var}(\mathrm{Attn}(\mathrm{LN}(x_\ell))) + \mathrm{Cov}(\mathrm{Attn}(\mathrm{LN}(x_\ell)), \mathrm{Var}(x_\ell)) \\ &= \sigma^2_{x_\ell} + \sigma^2_{\text{Attn}} + \rho_1 \cdot \sigma_{x_\ell} \cdot \sigma_{\text{Attn}},\end{aligned} \tag{13}$$

where ρ_1 is the correlation factor. Similarly, let $\mathrm{Var}(\mathrm{FFN}(\mathrm{LN}(x'_\ell))) = \sigma^2_{\mathrm{FFN}}$. Then we have:

$$\sigma^2_{x_{\ell+1}} = \sigma_(x'_\ell)^2 + \sigma^2_{\mathrm{FFN}} + \rho_2 \cdot \sigma_{x'_\ell} \cdot \sigma_{\mathrm{FFN}}, \tag{14}$$

where ρ_2 is the correlation factor. Thus, the relationship between $\mathrm{Var}(x_{\ell+1})$ and $\mathrm{Var}(x_\ell)$ becomes:

$$\sigma^2_{x_{\ell+1}} = \sigma^2_{x_\ell} + \sigma^2_{\text{Attn}} + \sigma^2_{\mathrm{FFN}} + \rho_1 \cdot \sigma_{x_\ell} \cdot \sigma_{\text{Attn}} + \rho_2 \cdot \sigma_{x'_\ell} \cdot \sigma_{\mathrm{FFN}}. \tag{15}$$

A.4.1 Variance of the Attention

The scaled dot-product attention mechanism is defined as:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

The softmax function outputs a probability distribution over the keys. Let the softmax output be $A = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$, where A is a matrix with each row summing to 1. The final attention output is obtained by multiplying the softmax output A with the value matrix V:

$$\mathrm{Attn}(Q, K, V) = AV.$$

To simplify the analysis, we make the following additional assumptions: The softmax output A is approximately uniform, meaning each element of A is roughly $1/n$, where n is the number of keys/values. Given this assumption, the variance of the attention is:

$$\mathrm{Var}(\mathrm{Attn}(Q, K, V)) \sim \mathrm{Var}(AV) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Var}(V_i) = \frac{1}{n} \cdot n \cdot \sigma^2_V = \sigma^2_V = \sigma^2_W. \tag{16}$$

where W is the universal weight matrix defined as before.

A.4.2 Variance of the Feed-Forward Network

The feed-forward network (FFN) in transformers typically consists of two linear transformations with a ReLU activation in between. The FFN can be written as:

$$\mathrm{FFN}(x) = W_2 \cdot \mathrm{ReLU}(W_1 \cdot x + b_1) + b_2. \tag{17}$$

where W_1 and W_2 are weight matrices, and b_1 and b_2 are bias vectors.

Using the result obtained by Wang et al. (2024), we get:

$$\sigma^2_{\mathrm{FFN}} \sim \sigma^2_{W_1} \cdot \sigma^2_{W_2} = \sigma^4_W. \tag{18}$$

In conclusion:

$$\begin{aligned}\sigma^2_{x'_\ell} &= \sigma^2_{x_\ell} + \sigma^2_W + \rho_2 \cdot \sigma_{x_\ell} \cdot \sigma_W \\ &= \sigma^2_{x_\ell}(1 + \frac{\sigma_W}{\sigma_{x_\ell}} + \frac{\sigma^2_W}{\sigma^2_{x_\ell}}) \\ &= \sigma^2_{x_\ell}\Theta(1 + \frac{1}{\sigma_{x_\ell}}).\end{aligned} \tag{19}$$

For simplicity, we set the numerator part to 1. Substitute $\sigma_{x'_\ell} = \sigma_{x_\ell}\sqrt{1+\frac{\sigma_W^2}{\sigma_{x_\ell}^2}+\rho_2\cdot\frac{\sigma_W}{\sigma_{x_\ell}}}$. into equation 15 we get:

$$\begin{aligned}\sigma_{x_{\ell+1}}^2 &= \sigma_{x_\ell}^2+\sigma_W^2+\sigma_W^4+\rho_1\cdot\sigma_{x_\ell}\cdot\sigma_W+\rho_2\cdot\sigma_{x'_\ell}\cdot\sigma_W^2\\ &= \sigma_{x_\ell}^2+\sigma_W^2+\sigma_W^4+\rho_1\cdot\sigma_{x_\ell}\cdot\sigma_W+\rho_2\cdot\sigma_{x_\ell}\cdot\sigma_W^2+\frac{\rho_2\sigma_W^4}{2\sigma_{x_\ell}}+\frac{\rho_2^2\sigma_W^3\sigma_{x_\ell}}{2}\\ &= \sigma_{x_\ell}^2\Theta(1+\frac{1}{\sigma_{x_\ell}}).\end{aligned} \tag{20}$$

From the result we can generally infer that the variance accumulates layer by layer. The variance with regard to σ_{x_1}:

$$\sigma_{x_\ell}^2 = \sigma_{x_1}^2\Theta\Big(\prod_{k=1}^{\ell-1}\Big(1+\frac{1}{\sigma_{x_k}}\Big)\Big). \tag{21}$$

We can also obtain a similar result for $\sigma_{x'_\ell}^2$.

We observe that for any $\sigma_{x_k}^2 \geq 1$, the sequence is increasing, meaning each term in the product is bounded. Consequently, the entire product is bounded above by:

$$\sigma_{x_\ell}^2 \leq \sigma_{x_1}^2\prod_{k=1}^{\ell-1}\Big(1+\sqrt{\frac{1}{\sigma_{x_1}}}\Big) = \sigma_{x_1}^2\big(1+\sqrt{\frac{1}{\sigma_{x_1}}}\big)^{\ell-1} = \exp\Theta(L). \tag{22}$$

Taking the natural logarithm of both sides:

$$\begin{aligned}\log(\sigma_{x_\ell}^2) &= \log\left(\sigma_{x_1}^2\prod_{k=1}^{\ell-1}\left(1+\sqrt{\frac{1}{\sigma_{x_k}^2}}\right)\right) = \sum_{k=1}^{\ell-1}\log\left(1+\sqrt{\frac{1}{\sigma_{x_k}^2}}\right)+\log(\sigma_{x_1}^2)\\ &\geq \sum_{k=1}^{\ell-1}\Big(\sqrt{\frac{1}{\sigma_{x_k}^2}}-\frac{1}{2}\left(\sqrt{\frac{1}{\sigma_{x_k}^2}}\right)^2\Big)+\log(\sigma_{x_1}^2).\end{aligned} \tag{23}$$

Exponentiating both sides to find the lower bound for $\sigma_{x_\ell}^2$, we obtain:

$$\sigma_{x_\ell}^2 \geq \sigma_{x_1}^2\exp\left(\sum_{k=1}^{\ell-1}\left(\sqrt{\frac{1}{\sigma_{x_k}^2}}-\frac{1}{2\sigma_{x_k}^2}\right)\right).$$

This provides a tighter lower bound for $\sigma_{x_\ell}^2$ compared to the upper bound of equation 22. Since we know the upper bound of variance grows exponentially, the lower bound must be sub-exponential. Therefore, for $\sigma_{x_\ell}^2 = \ell$, we must have:

$$\sigma_{x_\ell}^2 \geq \sigma_{x_1}^2\exp\left(\sum_{k=1}^{\ell-1}\left(\frac{1}{k}-\frac{1}{2k}\right)\right) = \Theta(\exp(\sqrt{L})) \geq \Theta(L).$$

Therefore, the increasing lower bound for $\sigma_{x_\ell}^2$ must grows faster than a linear function. So, the increasing of variance is sub-exponential.

□

A.5 Proof of Theorem 1

In this proof, we will divide the argument into two parts: first, the calculation of the Lemma 2, and second, the analysis of $\frac{\partial y_\ell}{\partial x_1}$.

Lemma 2. *For an L-layered Pre-LN Transformer, $\frac{\partial y_L}{\partial x_1}$ using Equations equation 2 and equation 3 is given by:*

$$\frac{\partial y_L}{\partial x_1} = \prod_{n=1}^{L-1} \left(\frac{\partial y_\ell}{\partial x'_\ell} \cdot \frac{\partial x'_\ell}{\partial x_\ell} \right). \tag{24}$$

The upper bound for the norm of $\frac{\partial y_L}{\partial x_1}$ is:

$$\begin{aligned} \left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq \prod_{l=1}^{L-1} \Big(\Big(1 + \frac{\sigma^2}{\sigma_{x'_\ell}(\sqrt{d} + \sqrt{d_{\text{FFN}}})^2} \Big) \times \Big(1 + 2dh \left(\sqrt{s} + 2 + \frac{1}{\sqrt{s}} \right) \frac{\sigma^2}{\sigma_{x_\ell}} \\ \times \Big(\sigma^2 d \sqrt{d_{\text{head}}} + \Big(1 + \sqrt{d_{\text{head}}/d} \Big) \Big) \Big) \Big). \end{aligned} \tag{25}$$

Here, h denotes the number of heads, s is the sequence length, and d, d_{FFN}, and d_{head} are the dimension of the embedding, FFN layer and multi-head attention layer, respectively. The standard deviation of W_Q, W_K, W_V, and W_{FFN} at layer ℓ is σ based on Assumption 1.

A.5.1 Proof of Lemma 2

Proof. Our derivation follows results in Takase et al. (2023), specifically Equation (7), which provides an upper bound on the norm of $\frac{\partial y_\ell}{\partial x_1}$ as:

$$\left\| \frac{\partial y_\ell}{\partial x_1} \right\|_2 = \left\| \prod_{l=1}^{L-1} \frac{\partial y_\ell}{\partial x'_\ell} \frac{\partial x'_\ell}{\partial x_\ell} \right\|_2. \tag{26}$$

Thus, we can estimate the upper bound of the gradient norm of $\frac{\partial y_\ell}{\partial x_1}$ by analyzing the spectral norms of the Jacobian matrices for the FFN layer and the self-attention layer, namely,

$$\text{FFN: } \left\| \frac{\partial y_\ell}{\partial x'_\ell} \right\|_2 \quad \text{Attention: } \left\| \frac{\partial x'_\ell}{\partial x_\ell} \right\|_2. \tag{27}$$

We now derive an upper bound of $\| \frac{\partial y_\ell}{\partial x'_\ell} \|_2$ as follows:

$$\left\| \frac{\partial y_\ell}{\partial x'_\ell} \right\|_2 \leq 1 + \left\| \frac{\partial \text{FFN}(\text{LN}(x'_\ell))}{\partial \text{LN}(x'_\ell)} \right\|_2 \left\| \frac{\partial \text{LN}(x'_\ell)}{\partial x'_\ell} \right\|_2. \tag{28}$$

Let σ_{w1_ℓ} and σ_{w2_ℓ} be the standard deviations of W^1_ℓ and W^2_ℓ, respectively. From Assumption 1, the spectral norms of W^1_ℓ and W^2_ℓ are given by their standard deviations and dimensions (Vershynin, 2018), so wo have:

$$\|W_1\|_2 \sim \sigma_1 \sqrt{d + \sqrt{d_{\text{FFN}}}}.$$

.

For simplicity, we assume that d, and d_{FFN} are equal, thus,

$$\left\| \frac{\partial \text{FFN}(\text{LN}(x'_\ell))}{\partial \text{LN}(x'_\ell)} \right\|_2 = \|W^1_\ell W^2_\ell\|_2 \leq \sigma_1 \sigma_2 (\sqrt{d} + \sqrt{d_{\text{ffn}}})^2. \tag{29}$$

Finally, we have the following bound:

$$\left\| \frac{\partial y_\ell}{\partial x'_\ell} \right\|_2 \leq 1 + \frac{\sigma_{w1_\ell} \sigma_{w2_\ell}}{\sigma_{x'_\ell}(\sqrt{d} + \sqrt{d_{\text{FFN}}})^2} = 1 + \frac{\sigma^2_\ell}{\sigma_{x'_\ell}(\sqrt{d} + \sqrt{d_{\text{FFN}}})^2}. \tag{30}$$

Following a similar procedure for the FFN, we rewrite $\| \frac{\partial x'}{\partial x} \|_2$ in equation 27 as:

$$\left\| \frac{\partial x'}{\partial x} \right\|_2 \leq 1 + \left\| \frac{\partial \text{Attn}(\text{LN}(x))}{\partial \text{LN}(x)} \right\|_2 \left\| \frac{\partial \text{LN}(x)}{\partial x} \right\|_2. \tag{31}$$

Let $Z(\cdot) = \text{concat}(\text{head}_1(\cdot), \ldots, \text{head}_h(\cdot))$ and J^Z denote the Jacobian of the $Z(\cdot)$. We can now express the spectral norm of the Jacobian matrix of attntion as:

$$\left\| \frac{\partial \text{Attn}(\text{LN}(x_\ell))}{\partial \text{LN}(x_\ell)} \right\|_2 = \left\| W_\ell^O Z(\text{LN}(x_\ell)) \frac{\partial Z(\text{LN}(x_\ell))}{\partial \text{LN}(x_\ell)} \right\|_2 = \|W_\ell^O J_\ell^Z\|_2. \tag{32}$$

From Vershynin (2018), we know that:

$$\|J_\ell^Z\|_2 \leq h\Big(\left(\sqrt{s} + 2 + \frac{1}{\sqrt{s}} \right) \sigma^3 \sqrt{d^3 d_{\text{head}}} + \sigma_x^\ell \left(\sqrt{d} + \sqrt{d_{\text{head}}} \right) \Big). \tag{33}$$

Here h is the number of heads, s is the sequence length, and the standard deviation of W_Q, W_K, and W_V is σ.

By combining the inequalities equation 30, equation 33 and equation 31, and assuming that all σ values are the same for simplicity. we obtain:

$$\begin{aligned} \left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq \prod_{l=1}^{L-1} \Big(\Big(1 + \frac{\sigma^2}{\sigma_{x'_\ell} (\sqrt{d} + \sqrt{d_{\text{FFN}}})^2} \Big) \times \Big(1 + 2dh \left(\sqrt{s} + 2 + \frac{1}{\sqrt{s}} \right) \frac{\sigma^2}{\sigma_{x_\ell}} \\ \times \Big(\sigma^2 d \sqrt{d_{\text{head}}} + \Big(1 + \sqrt{d_{\text{head}}/d} \Big) \Big) \Big). \end{aligned} \tag{34}$$

□

A.5.2 Analysis of the upper bound

As discussed in Takase et al. (2023), σ should be sufficiently small, and the standard deviation, $\sigma_{x'_\ell}$ or σ_{x_ℓ} should satisfy the condition $\sigma^2 \ll \sigma_{x'_\ell}$ to maintain the lazy training scheme. Thus, we obtain the following bound for the product over ℓ from 1 to L:

To find the bound for $\left\| \frac{\partial y_\ell}{\partial x_1} \right\|_2$ with respect to ℓ, we simplify the given inequality by approximating σ_{x_ℓ} and $\sigma_{x'_\ell}$. Based on equation 19, σ_{x_ℓ} is only one layer ahead of $\sigma_{x'_\ell}$, and this layer does not significantly affect the overall performance of deep Transformer networks. Furthermore, based on Lemma 1, we assume that $\sigma_{x'_\ell} = \sigma_{x_\ell}$.

equation 2 can be expressed in a traditional product form Whittaker & Watson (1996) for σ_{x_ℓ}:

$$\left\| \frac{\partial y_L}{\partial x_1} \right\|_2 \leq \prod_{l=1}^{L-1} \left(1 + \frac{1}{\sigma_{x_\ell}} A + \frac{1}{\sigma_{x_\ell}^2} B \right), \tag{35}$$

where

$$A = \frac{\sigma^2}{(\sqrt{d} + \sqrt{d_{\text{FFN}}})^2} + 2dh \left(\sqrt{s} + 2 + \frac{1}{\sqrt{s}} \right) \sigma^2 \left(d\sqrt{d_{\text{head}}} + 1 + \sqrt{d_{\text{head}}/d} \right), \tag{36}$$

and

$$B = 2dh \left(\sqrt{s} + 2 + \frac{1}{\sqrt{s}} \right) \sigma^4 d \sqrt{d_{\text{head}}}, \tag{37}$$

where A and B are independent of σ_{x_ℓ}, and under our assumption, are treated as constants.

From classical infinite series analysis, it is known that as σ_{x_ℓ} grows at a faster rate, the upper bound of the product decreases. The proof is omitted here for brevity. For the upper bound on the convergence rate of $\sigma_{x_\ell}^2$, we assume $\sigma_{x_\ell}^2 = \exp(\ell)$ without loss of generality. Under this condition, we can derive the following result:

Taking the natural logarithm of the product:

$$\log\left(\prod_{k=1}^{L-1}\left(1+\frac{A}{e^k}+\frac{B}{e^{2k}}\right)\right)=\sum_{k=1}^{L-1}\log\left(1+\frac{A}{e^k}+\frac{B}{e^{2k}}\right).$$

Using the Taylor series expansion for $\log(1+x)$, and applying this to our sum, we get:

$$\sum_{k=1}^{\infty}\log\left(1+\frac{A}{e^k}+\frac{B}{e^{2k}}\right)=\sum_{k=1}^{\infty}\left(\frac{A}{e^k}+\frac{B}{e^{2k}}-\frac{1}{2}\left(\frac{A}{e^k}+\frac{B}{e^{2k}}\right)^2+\frac{1}{3}\left(\frac{A}{e^k}+\frac{B}{e^{2k}}\right)^3-\cdots\right).$$

By evaluating the sums for each order of terms, we find that the result is a constant. Carrying this out for each term, we obtain:

$$\log\left(\prod_{k=1}^{L-1}\left(1+\frac{A}{e^k}+\frac{B}{e^{2k}}\right)\right)\sim\frac{A}{e-1}+\frac{B}{e^2-1}-\frac{1}{2}\left(\frac{A^2}{e^2-1}+2\frac{A\cdot B}{e^3-1}+\frac{B^2}{e^4-1}\right).$$

Thus, the product is approximately:

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\leq\exp\left(\frac{A}{e-1}+\frac{B}{e^2-1}-\frac{1}{2}\left(\frac{A^2}{e^2-1}+2\frac{A\cdot B}{e^3-1}+\frac{B^2}{e^4-1}\right)\right)=M,\tag{38}$$

where M is a constant.

For the lower bound on the convergence rate of $\sigma^2_{x_\ell}$, we assume $\sigma^2_{x_\ell}=\ell$ without loss of generality. Under this condition, we derive the following result. Taking the logarithm of the product, applying the Taylor series expansion for $\log(1+x)$, and applying this to our sum:

$$\sum_{k=1}^{\infty}\log\left(1+\frac{A}{k}+\frac{B}{e^{k^2}}\right)=\sum_{k=1}^{\infty}\left(\frac{A}{k}+\frac{B}{e^{k^2}}-\frac{1}{2}\left(\frac{A}{k}+\frac{B}{e^{k^2}}\right)^2+\frac{1}{3}\left(\frac{A}{k}+\frac{B}{e^{k^2}}\right)^3-\cdots\right).$$

For the first-order terms:

$$\sum_{k=1}^{\infty}\left(\frac{A}{k}+\frac{B}{e^{k^2}}\right)=A\sum_{k=1}^{\infty}\frac{1}{k}+B\sum_{k=1}^{\infty}\frac{1}{e^{k^2}}.$$

The series $\sum_{k=1}^{\infty}\frac{1}{k}$ is the harmonic series, which diverges. However, we approximate it using the Euler-Mascheroni constant γ and the fact recognize that the harmonic series grows logarithmically:

$$\sum_{k=1}^{\infty}\frac{1}{k}\sim\log n+\gamma\quad\text{(for large } n\text{)}.$$

The other series such as $\sum_{k=1}^{\infty}\frac{1}{e^{k^2}}$ converge because e^{k^2} grows very rapidly.

For higher-order terms, they converge to constant, involving the series $\sum_{k=1}^{\infty}\frac{1}{k^2}$ converges to $\frac{\pi^2}{6}$, so they contribute a constant. Exponentiating both sides, we get:

$$\prod_{k=1}^{\infty}\left(1+\frac{A}{k}+\frac{B}{e^{k^2}}\right)\sim\exp\left(A(\log n+\gamma)+const\right).$$

Thus, the growth rate of the upper bound for $\left\|\frac{\partial y_L}{\partial x_1}\right\|_2$ is:

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\leq\Theta(L).\tag{39}$$

B THEORETICAL ANALYSIS OF LAYERNORM SCALING

Lemma 3. *After applying our scaling method, the variances of x'_ℓ and x_ℓ, denoted as $\sigma^2_{x'_\ell}$ and $\sigma^2_{x_\ell}$, respectively, exhibit the same growth trend, which is:*

$$\sigma^2_{x_{\ell+1}} = \sigma^2_{x_\ell}\Theta(1 + \frac{1}{\sqrt{\ell}\sigma_{x_\ell}}), \tag{40}$$

with the following growth rate bounds:

$$\Theta(L) \leq \sigma^2_{x_L} \ \leq \ \Theta(L^{(2-\epsilon)}). \tag{41}$$

where ϵ is a small number with $0 < \epsilon \leq 1/4$.

From Lemma 3, we can conclude that our scaling method effectively slows the growth of the variance upper bound, reducing it from exponential to polynomial growth. Specifically, it limits the upper bound to a quadratic rate instead of an exponential one. Based on Theorem 1, after scaling, we obtain the following:

Theorem 2. *For the scaled Pre-LN Transformers, the Euclidean norm of $\frac{\partial y_L}{\partial x_1}$ is given by:*

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \prod_{\ell=1}^{L-1}\left(1 + \frac{1}{\ell\sigma_{x_\ell}}A + \frac{1}{\ell^2\sigma^2_{x_\ell}}B\right), \tag{42}$$

where A and B are dependent on the scaled neural network parameters. Then the upper bound for the norm is given as follows: when $\sigma^2_{x_\ell}$ grows at $\ell^{(2-\epsilon)}$, (i.e., at its upper bound), we obtain:

$$\sigma^2_{x_\ell} \sim \ell^{(2-\epsilon)}, \quad \left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \omega(1), \tag{43}$$

where ω denotes that if $f(x) = \omega(g(x))$, then $\lim_{x\to\infty}\frac{f(x)}{g(x)} = \infty$. Meanwhile, when $\sigma^2_{x_\ell}$ grows linearly (i.e., at its lower bound), we obtain:

$$\sigma^2_{x_\ell} \sim \ell, \quad \left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \Theta(L). \tag{44}$$

The detailed descriptions of A and B, and ϵ, along with the full proof, are provided in Appendices B.1 and B.2.

By comparing Theorem 1 (before scaling) with Theorem 2 (after scaling), we observe a substantial reduction in the upper bound of variance. Specifically, it decreases from exponential growth $\Theta(\exp(L))$ to at most quadratic growth $\Theta(L^2)$. In fact, this growth is even slower than quadratic expansion, as it follows $\Theta(L^{(2-\epsilon)})$ for some small $\epsilon > 0$.

When we select a reasonable upper bound for this expansion, we find that $\left\|\frac{\partial y_L}{\partial x_1}\right\|_2$ no longer possesses a strict upper bound. That is, as the depth increases, $\left\|\frac{\partial y_L}{\partial x_1}\right\|_2$ continues to grow gradually. Consequently, fewer layers act as identity mappings compared to the original Pre-LN where nearly all deep layers collapsed into identity transformations. Instead, the after-scaled network effectively utilizes more layers, even as the depth approaches infinity, leading to improved expressivity and trainability.

B.1 PROOF OF LEMMA 3

Proof. After scaling, the equation becomes:

$$\begin{aligned} y = x_{\ell+1} &= x'_\ell + \mathrm{FFN}(\frac{1}{\sqrt{\ell}}\mathrm{LN}(x'_\ell)), \\ x'_\ell &= x_\ell + \mathrm{Attn}(\frac{1}{\sqrt{\ell}}\mathrm{LN}(x_\ell)). \end{aligned} \tag{45}$$

Folloing the same analysis as before, we scale the Attention and FFN sub-layers, yielding:

$$\sigma^2_{\text{Attn}} = \frac{1}{n\ell} \cdot n \cdot \sigma_V^2 = \frac{1}{\ell}\sigma_V^2 = \frac{\sigma_W^2}{\ell}, \quad \sigma^2_{\text{FFN}} \sim \frac{\sigma^2_{W_1}}{\ell} \cdot \frac{\sigma^2_{W_2}}{\ell} = \frac{\sigma_W^4}{\ell^2}. \tag{46}$$

In conclusion:

$$\sigma^2_{x'_\ell} = \sigma^2_{x_\ell} + \sigma_W^2 + \rho_2 \cdot \sigma_{x_\ell} \cdot \frac{\sigma_W}{\sqrt{\ell}} = \sigma^2_{x_\ell}\Theta(1 + \frac{1}{\sqrt{\ell}\sigma_{x_\ell}}). \tag{47}$$

Similarly, we obtain:

$$\sigma^2_{x_{\ell+1}} = \sigma^2_{x_\ell}\Theta(1 + \frac{1}{\sqrt{\ell}\sigma_{x_\ell}}). \tag{48}$$

Taking the natural logarithm of both sides:

$$\begin{aligned} \log(\sigma^2_{x_\ell}) &= \log\left(\sigma^2_{x_1}\prod_{k=1}^{\ell-1}\left(1 + \sqrt{\frac{1}{\ell\sigma^2_{x_k}}}\right)\right) = \sum_{k=1}^{\ell-1}\log\left(1 + \sqrt{\frac{1}{\ell\sigma^2_{x_k}}}\right) + \log(\sigma^2_{x_1}) \\ &\geq \sum_{k=1}^{\ell-1}\Big(\sqrt{\frac{1}{\ell\sigma^2_{x_k}}} - \frac{1}{2}\left(\sqrt{\frac{1}{\ell\sigma^2_{x_k}}}\right)^2\Big) + \log(\sigma^2_{x_1}). \end{aligned} \tag{49}$$

To establish a lower bound for $\sigma^2_{x_\ell}$, we exponentiate both sides. Setting $\sigma^2_{x_\ell} = \ell$, we must have:

$$\sigma^2_{x_\ell} \geq \sigma^2_{x_1}\exp\left(\sum_{k=1}^{\ell-1}\left(\frac{1}{k} - \frac{1}{2k}\right)\right) = \Theta(\exp(\log L)) \geq \Theta(L). \tag{50}$$

Therefore, the increasing lower bound $\sigma^2_{x_\ell}$ is greater than a linear function.

Similarly, assuming $\sigma^2_{x_\ell} = \ell^{(2-\epsilon)}$, we have:

$$\sigma^2_{x_\ell} = \sigma^2_{x_1}\prod_{k=1}^{\ell-1}\left(1 + \frac{1}{\ell^{2-\epsilon/2}}\right) \sim \exp\left(\sum_{k=1}^{\ell-1}\frac{1}{k^{2-\epsilon/2}}\right) \sim \exp\left(\frac{\ell^{\epsilon/2-1} - 1}{\epsilon/2 - 1}\right) \leq \Theta(\ell^{(2-\epsilon)}) \leq \Theta(\ell^2). \tag{51}$$

Here ϵ is a small constant with $0 < \epsilon \leq 1/4$. Therefore, the increasing upper bound of $\sigma^2_{x_\ell}$ is slower than the ℓ^3 function, leading to:

$$\sigma^2_{x_\ell} \leq \Theta(L^2)$$

.

□

B.2 Proof of Theorem 2

Proof. Similarly, after applying the scaling transformation, we derive an upper bound for $\|\frac{\partial y_\ell}{\partial x'_\ell}\|_2$ as follows:

$$\begin{aligned} \left\|\frac{\partial y_\ell}{\partial x'_\ell}\right\|_2 &\leq 1 + \left\|\frac{\partial \text{FFN}(\text{LN}(x'_\ell))}{\partial \text{LN}(x'_\ell)}\right\|_2 \left\|\frac{1}{\sqrt{\ell}}\right\|_2 \left\|\frac{\partial \text{LN}(x'_\ell)}{\partial x'_\ell}\right\|_2 \\ &= 1 + \frac{\sigma_\ell^2}{\ell\sigma_{x'_\ell}(\sqrt{d} + \sqrt{d_{\text{FFN}}})^2}. \end{aligned} \tag{52}$$

Similarly, rewriting equation 27 after scaling, we have

$$\left\|\frac{\partial x'}{\partial x}\right\|_2 \leq 1 + \left\|\frac{\partial \text{Attn}(\text{LN}(x))}{\partial \text{LN}(x)}\right\|_2 \left\|\frac{1}{\sqrt{\ell}}\right\|_2 \left\|\frac{\partial \text{LN}(x)}{\partial x}\right\|_2. \tag{53}$$

By combining the bound equation 52, and inequality equation 53, and assuming all σ are equal for simplicity, we obtain:

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \prod_{l=1}^{L-1}\left(\left(1+\frac{\sigma^2}{\ell\sigma_{x'_\ell}(\sqrt{d}+\sqrt{d_{\text{FFN}}})^2}\right)\times\left(1+2dh\left(\sqrt{s}+2+\frac{1}{\sqrt{s}}\right)\frac{\sigma^2}{\ell\sigma_{x_\ell}}\right.\right.$$
$$\left.\left.\times\left(\sigma^2 d\sqrt{d_{\text{head}}}+\left(1+\sqrt{d_{\text{head}}/d}\right)\right)\right)\right). \tag{54}$$

equation 54 is a traditional product form Whittaker & Watson (1996) for σ_{x_ℓ}. After scaling, it becomes:

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \prod_{l=1}^{L-1}\left(1+\frac{1}{\ell\sigma_{x_\ell}}A+\frac{1}{\ell^2\sigma_{x_\ell}^2}B\right), \tag{55}$$

where A and B retain their forms from equation 36 and equation 37 and are treated as constants.

Regarding the upper bound on the convergence rate of $\sigma_{x_\ell}^2$, we assume $\sigma_{x_\ell}^2 = \ell^{(2-\epsilon)}$ without loss of generality. For large L, the product can be approximated using the properties of infinite products:

$$\prod_{\ell=1}^{L-1}\left(1+\frac{A}{\ell^{2-\epsilon/2}}+\frac{B}{\ell^{4-\epsilon}}\right) \sim \exp\left(\sum_{\ell=1}^{L-1}\left(\frac{A}{\ell^{2-\epsilon/2}}+\frac{B}{\ell^{4-\epsilon}}\right)\right). \tag{56}$$

Then, by evaluating the sum in the exponent, we obtain:

$$\prod_{\ell=1}^{L-1}\left(1+\frac{A}{\ell^{2-\epsilon/2}}+\frac{B}{\ell^{4-\epsilon}}\right) \sim \exp\left(A\cdot\frac{\ell^{\epsilon/2-1}-1}{\epsilon/2-1}+B\cdot\frac{\ell^{\epsilon-3}-1}{\epsilon-3}\right). \tag{57}$$

Therefore, we establish the upper bound:

$$\left\|\frac{\partial y_L}{\partial x_1}\right\|_2 \leq \Theta\left(\exp\left(A\cdot\frac{\ell^{\epsilon/2-1}-1}{\epsilon/2-1}+B\cdot\frac{\ell^{\epsilon-3}-1}{\epsilon-3}\right)\right) = \omega(1), \tag{58}$$

where $\omega(1)$ denotes a growth strictly greater than a constant as defined before. □

C Training Loss Curve

We report the training loss curve of Pre-LN and LayerNorm Scaling in Figure 4.

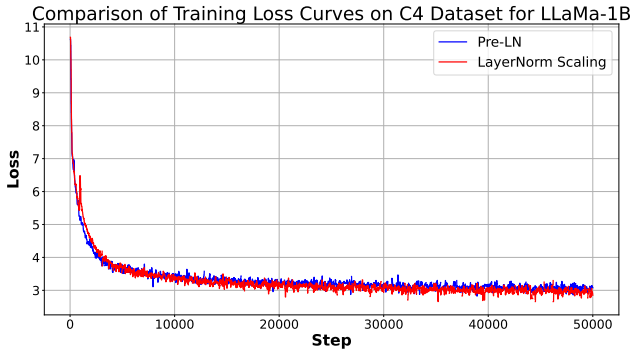

Figure 4: Training loss of LLaMA-1B with Pre-LN and LayerNorm Scaling.

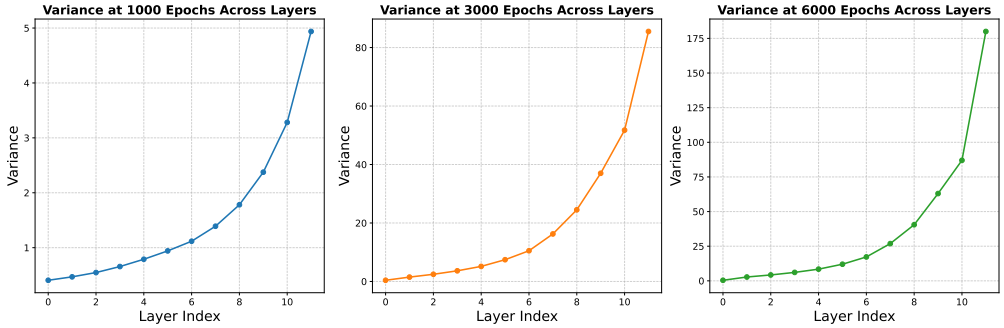

Figure 5: **Variance growth across layers in LLaMA-130M with Pre-LN.** Each subplot shows the variance at different training stages (1000, 3000, and 6000 epochs). In all cases, the variance follows an exponential growth pattern as depth increases, indicating that deeper layers experience uncontrolled variance amplification regardless of training progress.

D VARIANCE GROWTH IN PRE-LN TRAINING

To analyze the impact of Pre-LN on variance propagation, we track the variance of layer outputs across different depths during training.

Figure 5 illustrates the layer-wise variance in LLaMA-130M with Pre-LN at 1000, 3000, and 6000 epochs. Across all stages, variance remains low in shallow layers but grows exponentially in deeper layers, confirming that this issue persists throughout training rather than being a temporary effect. This highlights the necessity of stabilization techniques like LayerNorm Scaling to control variance and ensure effective deep-layer learning.