GPAS: Accelerating Convergence of LLM Pretraining via Gradient-Preserving Activation Scaling

Tianhao Chen[1*], **Xin Xu**[1*], **Zijing Liu**[2], **Pengxiang Li**[3], **Xinyuan Song**[4], **Ajay Kumar Jaiswal**[5], **Fan Zhang**[1], **Jishan Hu**[1], **Yang Wang**[1], **Hao Chen**[1], **Shizhe Diao**[6], **Shiwei Liu**[7], **Yu Li**[2], **Lu Yin**[8†], **Can Yang**[1†]
[1]The Hong Kong University of Science and Technology [2]International Digital Economy Academy
[3]Dalian University of Technology [4]Emory University [5]University of Texas at Austin
[6]NVIDIA [7]University of Oxford [8]University of Surrey
{tchenbb, xxuca}@connect.ust.hk, l.yin@surrey.ac.uk, macyang@ust.hk

Abstract

Modern Large Language Models, such as the LLaMA, Qwen and DeepSeek series, predominantly adopt the Pre-LayerNorm (Pre-LN) Transformer architecture. While being stable during pretraining and scalable to large model sizes, Pre-LN suffers from an exponential growth in activation variance across layers, causing the shortcut to dominate over sub-layer outputs in the residual connection and limiting the learning capacity of deeper layers. To mitigate this issue, we propose **Gradient-Preserving Activation Scaling** (GPAS), a simple technique that can be used in combination with existing approaches. GPAS works by scaling down the intermediate activations while keeping their gradients unchanged. This leaves information in the activations intact, and avoids the gradient vanishing problem associated with gradient downscaling. Extensive experiments across various model sizes from 71M to 1B show that GPAS achieves consistent performance gains. Beyond enhancing Pre-LN Transformers, GPAS also shows promise in improving alternative architectures such as Sandwich-LN and DeepNorm, demonstrating its versatility and potential for improving training dynamics in a wide range of settings. Our code is available at https://github.com/dandingsky/GPAS.

1 Introduction

The pursuit of more cost-effective and performant architectures is a central theme in the development of Large Language Models (LLMs). The introduction of the Transformer architecture [1] laid a solid foundation for modern language modeling and has since become the backbone of most state-of-the-art LLMs. To facilitate stable and scalable training, recent models such as the LLaMA, Qwen and DeepSeek series [2, 3, 4] adopt the Pre-LayerNorm (Pre-LN) Transformer architecture [5, 6, 7], enabling training models with up to hundreds of billions of parameters.

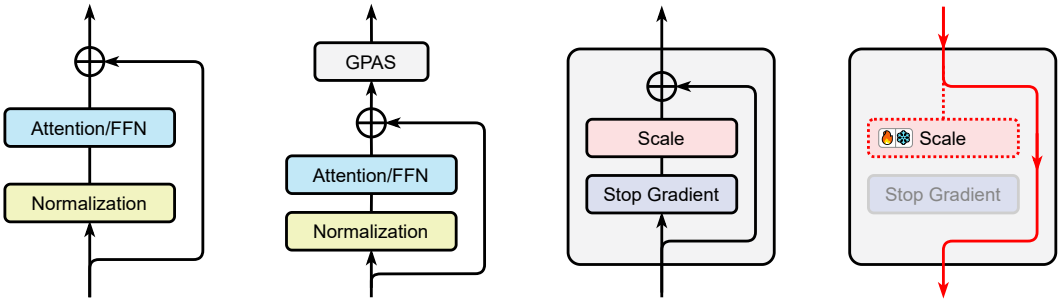
Although Pre-LN Transformers offer strong scalability and stability, they remain suboptimal in terms of convergence speed and parameter efficiency. Prior studies [8, 9, 10] have shown that deeper layers of Pre-LN Transformers can often be pruned or removed entirely with minimal impact on performance. While this creates opportunities for reducing inference costs, it also highlights a significant inefficiency in the training process—where most of the computational budget is spent.

One of the core reasons for the underutilization of deep layers lies in the accumulation of activation variance in Pre-LN Transformers. While the original Transformer architecture places Layer

* Equal contribution.
† Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Normalization [11] after the residual connection (commonly referred to as Post-LN), Pre-LN places it before the attention and FFN modules, leaving the shortcut branch unnormalized (Figure 1a). As a result, activation variance tends to accumulate with layer depth—often at an exponential rate. Recent works [12, 13] have shown that this variance growth causes the signal from attention and FFN modules to be increasingly overpowered by the shortcut branch, limiting the effectiveness of deeper layers in transforming hidden states.



(a) Pre-LN (b) Pre + GPAS (c) GPAS block (d) GPAS backprop

Figure 1: Architectures of Pre-LN, Pre + GPAS, and the GPAS block. Red lines indicate gradient flow. Red dotted line means the scale parameter can be either learnable or fixed.

In this work, we propose **Gradient-Preserving Activation Scaling** (GPAS), a simple yet effective solution to mitigate variance growth in Pre-LN Transformers. The idea of GPAS is straightforward: reduce variance growth by directly scaling down intermediate activations. However, naively scaling down activations in the forward pass leads to downscaled gradients during backpropagation, which could lead to vanishing gradients and slow down the learning process. As shown in Figure 1, GPAS works around this issue by scaling down forward activations while preserving the magnitude of backward gradients. Experiments on models ranging from 71M to 1B parameters show that Pre-LN with GPAS achieves faster convergence and improved downstream performance. Detailed analysis reveals that GPAS-augmented models exhibit a more compact distribution of activation variance across layers and more uniform layerwise importance.

Beyond enhancing Pre-LN Transformers, GPAS also serves as a general-purpose plugin for other architectural variants. We apply GPAS to architectures including DeepNorm [14], Sandwich-LN [15], Mix-LN [12] and LayerNorm Scaling [13], and observe consistent improvements in convergence speed and performance.

Our main contributions are summarized as follows:

- We propose Gradient-Preserving Activation Scaling to alleviate activation variance growth in Pre-LN Transformers while preserving gradient magnitudes.
- We validate the effectiveness of GPAS through pretraining experiments across various model sizes, and show that its benefits carry over to downstream supervised finetuning tasks.
- We further apply GPAS to other architectures such as DeepNorm, Sandwich-LN, Mix-LN and LayerNorm Scaling, and observe consistent performance gains.
- We conduct a thorough analysis of the training dynamics and model properties of models with and without GPAS.

2 Related Work

Normalization layers. Normalization is crucial for training deep neural networks, especially for modern LLMs with increasing depth and parameter count. Layer Normalization [11] plays a vital role in the success of the original Transformer [1]. Despite its effectiveness, improving normalization remains an active area of research. GroupNorm [16] divides channels into subgroups and normalizes within each group. RMSNorm [17], which omits mean centering and uses root mean square instead of variance, offers a simpler and more efficient alternative to LayerNorm, and has gained popularity in recent LLMs [2, 3, 4]. AdaNorm [18] proposed a new transformation function to replace the gain

and bias in LayerNorm. Dynamic Tanh [19] eliminates normalization altogether by replacing it with a non-linear activation.

LLM normalization schemes. Modern LLM architectures can be broadly characterized by three key components: normalization schemes, attention mechanisms, and FFN designs. Among these, normalization is most relevant to our work, while attention and FFN variants are largely orthogonal.

Transformer normalization has evolved around two main paradigms—Post-LN and Pre-LN, which have inspired numerous refinements to address their respective limitations. Post-LN was proposed in the vanilla Transformer [1], while Pre-LN was proposed in several works [5, 6, 7] to address the optimization issue of Post-LN. A theoretical perspective from [20] showed that Post-LN leads to larger gradients near the output layer, necessitating warm-up to avoid divergence. Conversely, Pre-LN yields smaller gradients in deeper layers, mitigating instability at initialization but also potentially curtailing the effectiveness of late-stage parameters. B2T [21] bypasses normalization in early layers to combat gradient decay of Post-LN. DeepNorm [14] modifies Post-LN by scaling up the residual connection before normalization and downscaling layer parameters during initialization. Sandwich-LN [15, 22] applies normalization both before and after the attention and FFN modules to further stabilize Pre-LN. Mix-LN [12] combines Post-LN and Pre-LN layers in the same model to alleviate both the variance growth of Pre-LN and the gradient vanishing issue of Post-LN. LayerNorm Scaling [13] scales output of normalization layers by inverse square root of layer depth, which facilitates a linear variance growth in Pre-LN.

As for attention and FFN architectures, existing variants mainly focus on expressiveness, scalability and hardware efficiency [23, 24, 25, 26]. While these directions are orthogonal to our work, we briefly mention them here to provide a more complete picture.

3 Gradient-Preserving Activation Scaling

3.1 Pre-LN with GPAS

We first demonstrate how to incorporate GPAS into the prevailing Pre-LN architecture, before extending to other baselines. Equation (1) shows a typical forward pass of a Pre-LN sub-layer (of layer l). GPAS introduces a simple modification after each sub-layer as Equation (2). Each layer now has an additional learnable scalar $\alpha_l \in \mathbb{R}$. $\mathrm{SiLU}(\cdot)$ is the Sigmoid Linear Unit [27]. $\mathrm{sg}(\cdot)$ is the stop gradient operator, which acts as an identity mapping during forward pass, but gives zero gradient during backpropagation. A visualization of this modification is shown in Figure 1.

The learnable gate α_l controls the amount of scaling that is applied to the intermediate activation x'_{l+1}. Moreover, by scaling x'_{l+1}, α_l controls the mixing ratio between shortcut x_{l+1} and the output of the next sub-layer $f(\mathrm{LN}(x_{l+1}))$, effectively balancing information from these 2 variables. SiLU activation is used to encourage positive α_l values to reduce activation variance, while also allowing the network to learn negative values when necessary. Notice that the input to the next sub-layer's attention or FFN is not scaled regardless of the value of α_l. This is because Pre-LN architecture applies LayerNorm right before the attention or FFN module, which cancels out any scaling applied to x'_{l+1}. Overall, Equation (2) scales activation by $(1 - \mathrm{SiLU}(\alpha))$, while the Jacobian $\partial x_{l+1}/\partial x'_{l+1} = I$.

$$\text{Pre-LN:} \quad x'_{l+1} = x_l + f(\mathrm{LN}(x_l)), \quad f \in \{\mathrm{Attn}, \mathrm{FFN}\}, \tag{1}$$

$$\text{+GPAS:} \quad x_{l+1} = x'_{l+1} - \mathrm{SiLU}(\alpha_l) \cdot \mathrm{sg}(x'_{l+1}). \tag{2}$$

Preserving gradient with stop gradient. We analyze the gradient preservation of GPAS using the first layer's gradient as example. For brevity only one sub-layer per layer is considered. Let $\beta_l = 1 - \mathrm{SiLU}(\alpha_l)$ be the scaling factor for each layer. If we replace $\mathrm{sg}(\cdot)$ with identity, Equation (2) becomes naive scaling: $x_{l+1} = x'_{l+1} \cdot \beta_l$. Suppose we have L transformer layers in total, with $\mathcal{L} = \mathcal{L}(x_L, y)$ being the loss. Then the gradient $\partial\mathcal{L}/\partial x_1$ is:

$$\text{naive scaling:} \quad \frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial \mathcal{L}}{\partial x_L} \prod_{l=1}^{L-1} \frac{\partial x'_{l+1}}{\partial x_l} \cdot \prod_{l=1}^{L-1} \beta_l,$$

$$\text{GPAS:} \quad \frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial \mathcal{L}}{\partial x_L} \prod_{l=1}^{L-1} \frac{\partial x'_{l+1}}{\partial x_l}.$$

Since $\beta_l < 1$, the product $\prod_{l=1}^{L-1} \beta_l$ decays exponentially with depth, causing gradient vanishing problem in early layers. Applying GPAS eliminates the scaling terms β_l during backpropagation and preserves gradient magnitude. Detailed derivations are provided in Appendix C.

3.2 Apply GPAS to Other Normalization Schemes

LayerNorm Scaling (LNS) [13] scales normalized output by inverse square root of layer depth with Equation (3). Since the scaling factor can be absorbed into the affine transformation inside LayerNorm, LNS is essentially Pre-LN with a different initialization of LayerNorm weights. Thus, we apply GPAS to LNS in the same way as Pre + GPAS.

$$\text{LNS:} \quad x'_{l+1} = x_l + f(\text{LN}(x_l)/\sqrt{l}), \quad f \in \{\text{Attn}, \text{FFN}\}. \tag{3}$$

$$\text{Sandwich-LN:} \quad x'_{l+1} = x_l + \text{LN}(f(\text{LN}(x_l))), \quad f \in \{\text{Attn}, \text{FFN}\}. \tag{4}$$

For Sandwich-LN [15], GPAS also applies after the residual connection with Equation (2). The only difference from Pre-LN is that the sub-layer forward has an additional LayerNorm as shown in Equation (4).

For Post-LN and DeepNorm [1, 14], however, the scenarios are quite different. As shown in Equation (5), Post-LN breaks the residual connection by wrapping $x_l + f(x_l)$ with LayerNorm. DeepNorm inherits this architecture and introduces a scaling factor α to the shortcut x_l and another scaling factor β to the sub-layer weights in f, where both α and β are predefined and fixed during training. Empirically, we found applying GPAS after LayerNorm does not bring performance gain. Instead, we hypothesize that the scaling factor α is not optimal since it does not account for layerwise variation and training dynamics. Therefore, we apply GPAS to the scaled shortcut $\alpha \cdot x_l$, while keeping the input to attention and FFN unscaled. Ultimately, we found applying GPAS as Equation (7) and (8) boosts pretrain performance.

$$\text{Post-LN:} \quad x_{l+1} = \text{LN}(x_l + f(x_l)), \quad f \in \{\text{Attn}, \text{FFN}\}. \tag{5}$$

$$\text{DeepNorm:} \quad x_{l+1} = \text{LN}(\alpha \cdot x_l + f_\beta(x_l)), \quad f_\beta \in \{\text{Attn}, \text{FFN}\}. \tag{6}$$

$$\text{DeepNorm + GPAS:} \quad x'_l = x_l - \text{SiLU}(\alpha_l) \cdot \text{sg}(x_l), \tag{7}$$

$$x_{l+1} = \text{LN}(\alpha \cdot x'_l + f_\beta(x_l)), \quad f_\beta \in \{\text{Attn}, \text{FFN}\}. \tag{8}$$

Mix-LN [12] combines Pre-LN and Post-LN layers in the same model, so we apply GPAS differently to its Pre- and Post-LN layers. For the Post-LN layers, we apply the same strategy as Equation (7). For the Pre-LN layers, we use Equation (2).

4 Experiments and Results

4.1 Experiment Setup

Model architectures. Based on our proposed architectural modifications in Section 3, we conduct extensive experiments to verify their effectiveness. Following [28] and [29], we employ LLaMA-based model architectures for implementing Post-LN [1], DeepNorm [14], Pre-LN [30], Sandwich-LN [31], Mix-LN [12] and LNS [13]. All models share the same attention and FFN architectures as well as normalization layers except for normalization scheme. Specifically, all baseline architectures (from Post-LN to LNS) utilize RMSNorm [17], LLaMA attention and LLaMA MLP [32] with SwiGLU activation [33]. Moreover, they share the same initialization except for DeepNorm and LNS, which add additional scaling to certain parameters. Scaled Embed [34] is applied to stabilize pretraining. We then apply GPAS to these baseline architectures according to Section 3, and refer to the modified architectures as *DeepNorm + GPAS*, *Pre + GPAS*, *Sandwich + GPAS*, *Mix + GPAS*, and *LNS + GPAS*, respectively. Notice that we did not conduct experiment on Post + GPAS, since we found DeepNorm to be a better starting point in a preliminary study, and utilized our compute budget on DeepNorm + GPAS instead.

Pretraining. We perform pretraining experiments across all architectures and at five model scales: 71M, 130M, 250M, 350M, and 1B. For the larger 7B configuration, we only pretrain Pre-LN and Pre + GPAS due to limited computational resources. Following [28], we adopt the Adam [35] optimizer and train on the C4 dataset [36, 37]. We tokenize the pretraining corpus with T5 tokenizer [36]

since it was trained on the C4 dataset. **For models with GPAS, we initialize all learnable gates $\alpha_l = 0$.** When $\alpha_l = 0$, GPAS does not scale the activations or alter the gradients, meaning the very first forward and backward passes during training are identical to those of the baseline model. We did not explore alternative initialization schemes for α_l. We also use the same gate value for both attention and FFN sub-layers within the same layer.

All experiments are carried out on 4 $\times$ NVIDIA H800 GPUs. Models below 350M can be trained in under 1 to 8 hours, while each 1B model took 2 to 3 days of training. More detailed configurations are listed in Table 1.

Table 1: Pretraining configurations for models of different sizes.

Model Size	Learning Rate	Warmup Steps	Training Steps	Batch Size	Train Tokens	Eval Tokens
71M	1×10^{-3}	1K	10K	512	1B	10M
130M	1×10^{-3}	2K	20K	512	2B	10M
250M	1×10^{-3}	4K	40K	512	4B	10M
350M	5×10^{-4}	6K	60K	512	6B	10M
1B	5×10^{-4}	10K	100K	512	10B	10M

Supervised finetuning. To determine whether the improvements observed during pretraining persist in subsequent training stages, we perform SFT on the pretrained models with 1B parameters from Section 4.2, and then evaluate their performance on common reasoning benchmarks. Following [12], we finetune the models on the Commonsense170K dataset [38] and evaluate the models on seven downstream tasks. **The learnable gates α_l are frozen during SFT to avoid disturbing pretrained knowledge.** We use a learning rate of 3×10^{-4} and train for 4 epochs using LISA [39]. As for evaluation, we adopt the widely used LM Evaluation Harness [40].

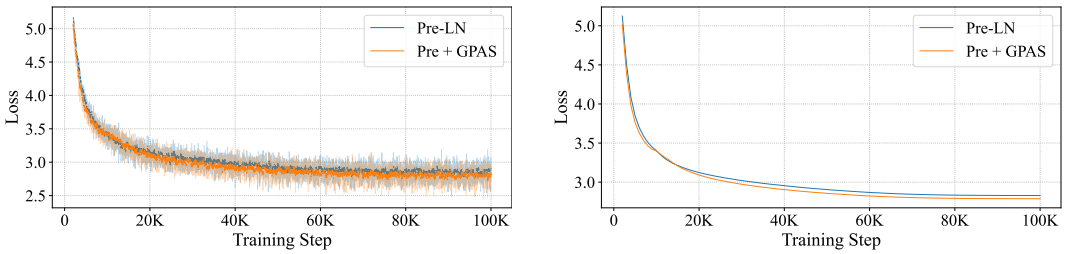
4.2 Pretrain Results

Table 2: Perplexity (↓) of pretrained models on evaluation set. Numbers in parentheses show improvement over the base method.

Method	71M	130M	250M	350M	1B
Post-LN [1]	33.80	26.50	1351.58	21.19	1406.66
DeepNorm [14]	35.49	26.78	22.20	21.76	1400.39
DeepNorm + GPAS	34.78 (-0.71)	26.62 (-0.16)	21.89 (-0.31)	21.29 (-0.47)	16.01 (-1384)
Pre-LN [20]	33.98	26.61	21.54	20.71	16.53
Pre + GPAS	33.38 (-0.60)	26.25 (-0.36)	21.34 (-0.20)	19.77 (-0.94)	16.11 (-0.42)
Sandwich-LN [15]	32.28	25.31	20.43	20.20	16.26
Sandwich + GPAS	**31.44** (-0.84)	**24.86** (-0.45)	20.38 (-0.05)	19.45 (-0.75)	15.85 (-0.41)
Mix-LN [12]	33.88	26.29	21.52	20.73	15.87
Mix + GPAS	33.26 (-0.62)	26.03 (-0.26)	21.43 (-0.09)	19.82 (-0.91)	15.38 (-0.49)
LNS [13]	34.58	25.91	20.59	20.35	15.61
LNS + GPAS	32.68 (-1.90)	24.95 (-0.96)	**19.89** (-0.70)	**19.38** (-0.97)	**14.87** (-0.74)

Table 2 summarizes the pretraining results across all model sizes and architectures. Results for 7B-parameter Pre-LN and Pre + GPAS are provided in Appendix B. We have the following observations: ❶ **Consistent Gains from GPAS.** For nearly all architectures and model sizes, applying GPAS leads to lower perplexities (numbers in parentheses), confirming that preserving gradient magnitudes while adjusting activation scales is beneficial. ❷ **Best Baselines at Different Scales.** At smaller scale (71M), Sandwich-LN is the strongest baseline (32.28), and adding GPAS further reduces perplexity to 31.44. At larger scale (1B), LNS already outperforms other baselines (15.61), but GPAS still offers a notable improvement (14.88). ❸ **Magnitude of Improvement.** The improvements (in parentheses) range from moderate ($\approx$ 0.3–0.7 drop in perplexity) to quite substantial (e.g., -1.90 for LNS at 71M). These consistent gains highlight that GPAS can be seamlessly integrated with various normalization schemes and model sizes, providing an effective and lightweight solution to boost convergence and performance in LLM pertaining.

For a detailed analysis on how GPAS enhances performance, please refer to Section 5.



(a) Pretrain loss

(b) Evaluation loss

Figure 2: Pretrain and evaluation loss of Pre-LN and Pre + GPAS, 1B parameters

Table 3: 0-shot performance (↑) on various benchmarks (1B-parameter models). All numbers are accuracy in %. **Bold**: higher among GPAS and baseline method; ***Bold italic***: highest among all.

Method	MMLU	BoolQ	PIQA	SIQA	HellaSwag	WinoG	ARC-e	ARC-c	OBQA	Average
Post-LN	22.95	37.83	52.77	34.03	26.20	48.15	27.36	19.37	11.40	31.12
DeepNorm	22.95	37.83	52.77	34.08	26.20	51.14	27.31	19.37	11.40	31.45
DeepNorm + GPAS	**26.46**	***62.11***	**69.53**	**46.93**	**34.37**	**52.09**	**49.24**	**22.61**	**20.40**	**42.64**
Pre-LN	25.96	50.34	68.66	44.27	32.39	51.14	49.37	21.33	17.60	40.12
Pre + GPAS	**26.68**	**59.79**	**69.31**	**46.52**	**33.64**	**52.49**	**49.79**	**22.70**	**22.00**	**42.55**
Sandwich-LN	**27.42**	61.77	67.63	44.68	32.76	**50.67**	47.43	23.12	21.40	41.88
Sandwich + GPAS	27.29	**61.90**	**69.15**	**45.29**	**34.61**	50.36	**51.39**	**23.46**	**22.20**	**42.85**
Mix-LN	**26.24**	61.93	68.66	45.50	33.09	52.25	48.78	**24.40**	20.80	42.40
Mix + GPAS	26.23	**61.99**	**69.59**	**45.60**	**33.51**	***53.51***	**50.34**	22.35	**22.40**	**42.83**
LNS	26.62	**62.02**	69.48	45.39	34.76	**51.38**	50.88	23.29	19.80	42.63
LNS + GPAS	***27.78***	61.56	***71.00***	***47.49***	***36.19***	51.22	***52.57***	***25.51***	***24.40***	***44.19***

4.3 SFT Results

We present the finetuning results in Table 3 and notice the following observations: ❶ **Gains from GPAS Across Architectures.** When GPAS is applied (bottom half), every architecture sees a measurable boost in average accuracy. For example, Pre + GPAS achieved a notable 2.43% increase in average accuracy. DeepNorm + GPAS achieves the most dramatic improvement since vanilla DeepNorm diverged during pretraining, while incorporating GPAS successfully avoided this catastrophic behavior. This suggests that DeepNorm's fixed residual scaling may not be optimal yet for network stability, and GPAS automatically learns to increase stability and convergence speed. ❷ **Task-Specific Improvements.** Breaking down the gains by individual datasets shows that the enhancements are typically consistent across tasks but vary in magnitude. For instance, Pre + GPAS's performance on BoolQ reached a 9.45% increase in accuracy, OBQA also has a notable gain of 4.4%. Meanwhile, LNS + GPAS attains improvements in both OBQA (+4.60) and HellaSwag (+1.43), pushing its overall average to 44.19% (+1.56).

Overall, by adaptively controlling the scale of the activations while preserving gradient magnitudes, GPAS consistently unlocks additional gains in downstream tasks.

5 Analysis

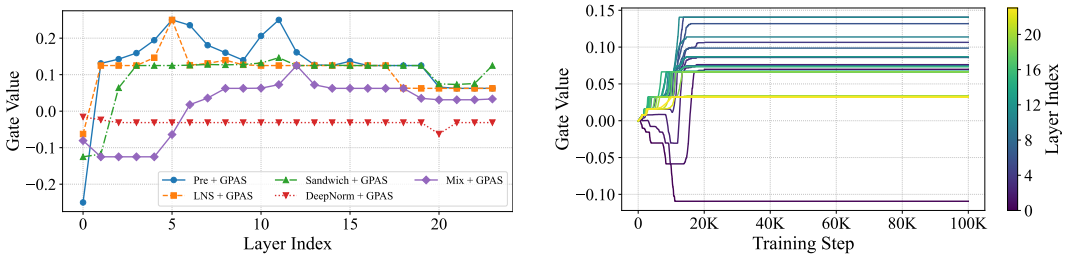
In this section, we provide a thorough analysis of GPAS by examining its training dynamics and model properties. We base our analysis primarily on Pre-LN and its GPAS-augmented version with 1B parameters. We also provide a detailed ablation study in Appendix A.

5.1 Learned Layerwise Gate Values

GPAS introduces a learnable gate α_l for each layer. We visualize the learned values of α_l in Figure 3. All models are the 1B-parameter models trained in Section 4.2. Generally, Pre-LN and Sandwich-LN layers tend to learn positive gates, while Post-LN layers tend to learn negative gates. Notably, for the Mix + GPAS model, the first 6 Post-LN layers learned negative gates, and the remaining Pre-LN layers learned positive values.

Across all model variants, the first layer tend to learn negative gate values. This means the network scales up the activation in the first layer. We hypothesize that this is due to the low variance of the initial word embeddings, and the network automatically learns to scale up that embedding to match the variance of subsequent layers. This hypothesis is also supported by Figure 4b, where the input activation to the first layer (which is the initial word embedding) has a much lower variance than the remaining layers.

Throughout training, the gates α_l exhibit the most variation during the first 10 – 20% of steps. After that, they tend to remain relatively stable. One likely reason for this behavior is the use of half-precision training with BFloat16, which may suppress small updates to the gate values due to limited numerical precision.



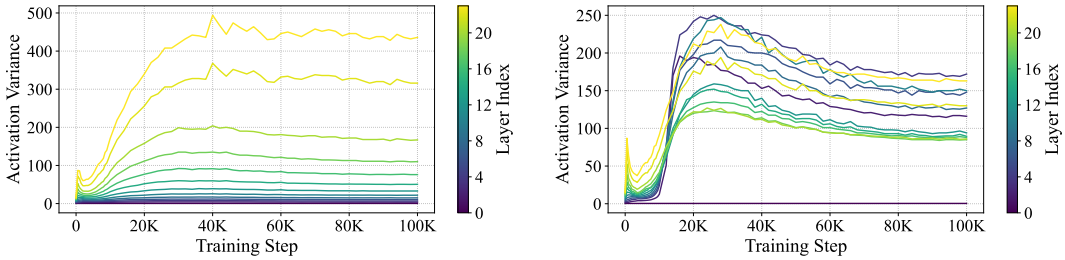
(a) Learned gate values α_l for different models

(b) Activated gate values $\mathrm{SiLU}(\alpha_l)$ across training steps of Pre + GPAS

Figure 3: Learned layerwise gate values

5.2 Activation Variance Comparison

We compare the activation variance of Pre-LN and Pre + GPAS, across all pretraining steps and layers. Experiments are conducted on 1B parameter models. We only plot every 2 layers including the first and last layers for brevity. Figure 4 shows that Pre-LN has an exponential increase in activation variance across layers, while Pre + GPAS offers a near 50% decrease in highest activation variance, with a more uniform and compact activation variance distribution across layers.



(a) Pre-LN

(b) Pre + GPAS

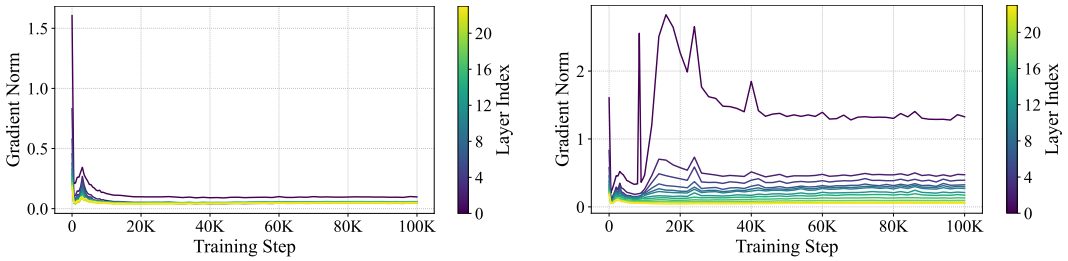
Figure 4: Layerwise activation variance with and without GPAS

5.3 Gradient Norm Comparison

We verify whether GPAS preserves gradient while scaling down activations. Figure 5 shows the layerwise gradient norms across across training steps. Models are the 1B parameter variants we trained in Section 4.2. It's evident that Pre + GPAS has a much larger gradient compared to its baseline method. Most layers in Pre-LN have gradient norm around 0.05 – 0.1, while Pre + GPAS's are scaled to 0.05 – 0.5. Although the first layer's gradient of Pre + GPAS seemed overly aggressive, the model was able to adjust to that gradient scale and achieve faster convergence.

However, we did find that the gradient spike around step 10K in Figure 5b disturbed the pretraining process. The gate values at step 10K went through an aggressive fluctuation (see Figure 3b), which caused the model to adjust to that change at steps 10K – 30K. This is also observed in the loss curves

in Figure 2, where both pretrain and evaluation loss of Pre + GPAS temporarily went above that of Pre-LN around step 10K. One simple way to fix this issue is to use gradient clipping on the gate values α_l. However, we found that, while stablizing training, applying an additional gradient clipping to the gate values slightly degrades final performance (PPL 16.11→16.21). This suggests that the gate values might need an update scheme different from the main parameters. For our main experiments, we apply gradient clipping of 1.0 to all parameters except for the gates. For other GPAS-augmented baselines, we did not encounter such fluctuation in gate values which disturbs training.

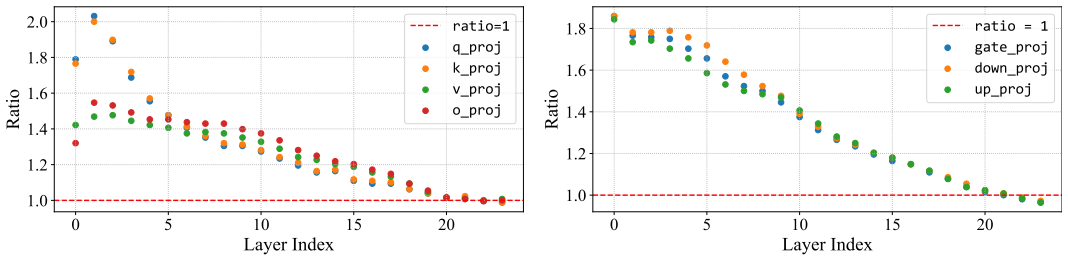


(a) Pre-LN (b) Pre + GPAS

Figure 5: Layerwise gradient norm with and without GPAS

5.4 Weight Norm Comparison

Since Pre + GPAS scales up activations in the early layers according to Figure 4b, we suspect that its early layer weights also have larger norms. Specifically, we compare the norm of weights in the attention and FFN modules. Figure 6 shows that Pre + GPAS has larger weights in almost all layers compared to Pre-LN, especially in the early layers where activation variances are relatively small. This effectively scales up the output variance of attention and FFN modules, such that early layers and deeper layers share similar activation variance. We believe that the combination of larger weights and stable activation variance across layers enables the network to tolerate larger gradients without crashing, which in turn speeds up convergence.



(a) Attention weight norm ratio (b) FFN weight norm ratio

Figure 6: Attention and FFN weight norm ratios of Pre + GPAS over Pre-LN.

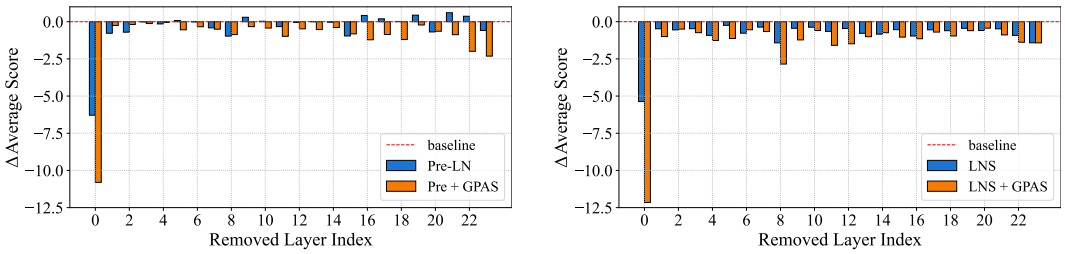
5.5 Layer Importance Comparison

We compare the layerwise importance of models with and without GPAS. The importance of a given layer l is measured as the performance drop after removing that layer. We take the finetuned models in Section 4.3 as baseline, and measure the difference in average score across those benchmarks listed in Table 3 after removing each one of the layers.

Figure 7a shows that Pre + GPAS increases layer importance for most and especially the deeper layers, while some layers in vanilla Pre-LN are even harmful to performance. This indicates that GPAS enables a more efficient utilization of model parameters, while Pre-LN's exponential variance growth limited the learning capacity of deeper layers.

We also show layer importance for LNS + GPAS in Figure 7b. LNS mitigates variance growth in Pre-LN by downscaling LayerNorm output with square root of layer depth. In this case, GPAS still

amplifies the importance of each layer by a small but noticeable amount, leading to a higher overall performance.



(a) Pre-LN vs. Pre + GPAS

(b) LNS vs. LNS + GPAS

Figure 7: Layer importance measured by performance drop after removal.

5.6 Gradient Analysis

Based on our experimental findings, we provide a brief and preliminary analysis to the activation variance and gradient profiles of Pre-LN and Pre + GPAS. By the result of [13] Theorem 1 and Lemma 1, and the work of [34], we have the following result for Pre-LN transformers:

Lemma 1 (Variance and Gradient Growth in Pre-LN)**.** *For a Pre-LN Transformer with L layers, using Equations* (1)*, and let W_ℓ be the model parameter matrix at layer ℓ. Assume that for all layers, x_ℓ, x'_ℓ, and W_ℓ are mutually independent and follow Gaussian distributions with mean zero. Let y_L be the output of the L-th layer, and consider the gradient norm $\left\|\frac{\partial y_L}{\partial x_1}\right\|_2$. Let the upper bound for this norm denoted by* $\mathrm{UP}(\cdot)$*. Then it satisfies:*

$$M \leq \mathrm{UP}\left(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\right) \leq O(L), \tag{9}$$

When we add the GPAS to the Pre-LN transformers, we have the following theorem

Theorem 1 (Variance and Gradient Growth in Pre-LN with GPAS)**.** *Using Equations* (1) *and* (2)*, and under the same assumptions as in Lemma 1, assume that each α_ℓ is bounded and varies slowly across layers. Let $L(\alpha)$ and $M(\alpha)$ denote the layerwise lower and upper bounds of* $\log(1-\mathrm{SiLU}(\alpha_\ell))$*, respectively. Let σ denote the variance of x_ℓ. Then the upper bound of the gradient norm, denoted as* $\mathrm{UP}\left(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\right)$*, satisfies the following inequality:*

$$\exp\left\{O\left(\frac{1}{\sigma}\frac{1}{1-e^{(-\frac{1}{\sigma+1}+L(\alpha))/2}}+1\right)\right\} \leq \mathrm{UP}\left(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\right) \leq \exp\left\{O\left(\exp\left\{-\frac{1}{2}(M(\alpha))\right\}\frac{\min\{L,\sigma\}}{\sigma}\right)\right\} \tag{10}$$

The detailed description as well as the complete proof, are provided in Appendix D. From Theorem 1, By comparing Lemma 1 (before GPAS) with Theorem 1 (after GPAS), For the lower bound of the gradient norm upper estimate $\mathrm{UP}(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2)$, the term $L(\alpha)$ effectively acts as a compensatory factor that offsets the influence of σ, thereby accelerating the growth of the bound beyond a constant rate. This leads to a strictly non-constant lower bound, which in turn ensures that the gradient magnitude retains sufficient variability across layers. In particular, this precludes the possibility of vanishing gradients and encourages meaningful gradient flow during backpropagation.

In contrast, for the upper bound, the term $M(\alpha)$ serves as a multiplicative scaling factor that exponentially suppresses the bound. This results in a significantly lower gradient norm ceiling compared to the $O(L)$ upper bound of standard Pre-LN Transformers. Consequently, the proposed modification mitigates gradient explosion and reduces the occurrence of large loss spikes, thereby enhancing training stability. Notably, the improved upper bound grows more slowly than L, further dampening excessive gradient amplification and promoting smoother optimization dynamics.

6 Conclusion

We proposed Gradient-Preserving Activation Scaling, a simple method that mitigates the exponential growth of activation variance in Pre-LN Transformers, thereby accelerating convergence and enhancing parameter efficiency. Beyond improving Pre-LN Transformers, GPAS also proves to be a viable plugin for alternative normalization schemes such as DeepNorm, Sandwich-LN, Mix-LN, and LNS.

Limitations. While GPAS showed consistent gains in pretraining performance, our experiments are limited to 1B-parameter models due to computational constraints. Additionally, the current use of the SiLU activation may still permit unstable gate updates that disrupt training dynamics—although gradient clipping partially mitigates this issue. Nevertheless, the learnable gates introduce uncertainty during pretraining, and a predefined gate schedule with theoretical guarantees might be more preferable, especially at larger scales. A deeper understanding of how GPAS affects training stability and convergence remains an open question. Finally, GPAS is intended for training LLMs from scratch, and applying it to models pretrained without GPAS will likely be suboptimal.

Broader Impact. GPAS offers a lightweight solution to improve the convergence speed and parameter efficiency of LLM pretraining. This may contribute to more accessible and sustainable development of foundation models, with potential downstream benefits across applications such as education and scientific research.

Acknowledgments

This work was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Reference Number: AoE/E-601/24-N).

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

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

[3] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[4] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[5] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *ICLR*, 2019.

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[7] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *ACL*, 2019.

[8] 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. *ICML*, 2024.

[9] 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.

[10] 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.

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

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

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

[14] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *TPAMI*, 2024.

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

[16] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, pages 3–19, 2018.

[17] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *NeurIPS*, 32, 2019.

[18] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *arXiv preprint arXiv:1911.07013*, 2019.

[19] Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization, 2025.

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

[21] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. B2t connection: Serving stability and performance in deep transformers. *ACL*, 2023.

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

[23] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

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

[25] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[26] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[27] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[28] Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *ICLR*, 2023.

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

[30] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ACL*, 2019.

[31] Xinyu Gong, Wuyang Chen, Tianlong Chen, and Zhangyang Wang. Sandwich batch normalization: A drop-in replacement for feature distribution heterogeneity. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2494–2504, 2022.

[32] 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.

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

[34] 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.

[35] Diederik P Kingma. Adam: A method for stochastic optimization. *ICLR*, 2015.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[37] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021.

[38] 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. *EMNLP*, 2023.

[39] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. 2024.

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

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

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

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We proposed GPAS which acclerates convergence and boosts performance of LLM pretraining, extensive experiments in Sections 4 and 5 confirms the utility of our method.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitations of our work in Sections 4, 5 and 6, addressing drawbacks such as limited model scale and potential training instability issues.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide proofs of our theoretical results (Section 5.6) in Section D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided details of our training approach in Sections 4 and 5, covering model architecture, hyperparameter, additional techniques, etc. We also provide code to replicate our experiments in the supplementary files of this submission.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: In Section 4, we provided citations to the open-source training framework and dataset that we used with our experiments, along with training configurations in Table 1. We also provide code to replicate our experiments in the supplementary files of this submission.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide details of our training settings in Sections 4, 5. We also provide code to replicate our experiments in the supplementary files of this submission.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Running multiple pretraining experiments to produce error bars is very costly and time-consuming, and out of our computational budget. However, we do run our proposed method across various baselines to show that our method consistently improves all baseline methods (Table 2 and 3).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided details on compute resources in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We made sure to follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential impacts of our work in Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our models are trained on the well established C4 dataset, and there's little safety concern since the generative ability of our pretrained models are limited by model and data scale.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide license information of assets used in our paper in Section E.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We only use LLMs for writing at the sentence level.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Ablation Study

We perform a series of ablation studies on GPAS to elucidate the rationale behind its architectural design. In particular, we investigate the impact of four key factors: the *Activation function*, the *Application location of GPAS*, the *Necessity of stop gradient operator*, and the choice between *Learnable vs. predefined gate value*. All ablations are conducted on 350M-parameter models, chosen because this scale typically serves as a reliable proxy for larger models while remaining feasible within our computational budget. We also limit the baseline architecture to Pre-LN only, and leave explorations of other architectures to future research.

Activation function. GPAS uses the SiLU activation by default. To enable more customizable control over activated gate values, we generalize GPAS to Equation (11), where $\mathrm{Act}(\cdot)$ can be any activation function. We then examine how different activation functions affect performance, using a 350M-parameter Pre-LN model with GPAS. Table 4 shows that $\mathrm{Identity}$ and Tanh achieve slightly lower perplexities than SiLU in the 350M setting. However, this advantage does not persist at larger scale: in the 1B setting, $\mathrm{Identity}$ and SiLU achieve perplexities of 16.49 and 16.25, respectively. We hypothesize that these activations, particularly $\mathrm{Identity}$, permit overly aggressive gate updates during training; by contrast, SiLU imposes a smoother gradient profile, constraining updates to a more stable range.

$$x_{l+1} = x'_{l+1} - \mathrm{Act}(\alpha_l) \cdot \mathrm{sg}(x'_{l+1}). \tag{11}$$

Table 4: Ablations on activation function (left) and GPAS insertion position (right). Based on 350M Pre-LN. SiLU ($\beta = 8$) refers to scaled SiLU: $x \cdot \mathrm{sigmoid}(\beta x)$.

Activation	Perplexity
Identity	20.08 (**-1.27**)
ReLU	21.17 (-0.18)
LeakyReLU	21.17 (-0.18)
Tanh	20.09 (-1.26)
SiLU ($\beta = 8$)	20.26 (-1.09)
SiLU (default)	20.35 (-1.00)
No GPAS	21.35

GPAS Position	Perplexity
After sub-layer (default)	20.35 (**-1.00**)
Before sub-layer	20.73 (-0.62)
After LayerNorm	21.33 (-0.02)
After Attn / FFN	21.23 (-0.12)
No GPAS	21.35

Where to apply GPAS. Since GPAS can, in principle, be integrated at any point where intermediate activations arise, we investigate the impact of inserting it at different locations within the transformer block. In Table 4, we compare four variants against a 350M-parameter Pre-LN baseline: (1) our default setting, which applies GPAS right after the sub-layer (residual + module output); (2) applying GPAS before the sub-layer; (3) placing it immediately after the LayerNorm; and (4) after each Attn/FFN module. While each insertion strategy outperforms the no-GPAS baseline, the default approach (inserting GPAS after the sub-layer) offers the largest perplexity reduction (-1.00), indicating that modulating the combined residual and module output is the most effective choice.

Necessity of stop gradient operator. We conduct a control experiment to show the necessity of the stop gradient operator in GPAS. For the control experiment, we use Equation (12) to replace Equation (2):

$$x_{l+1} = x'_{l+1} - \mathrm{SiLU}(\alpha_l) \cdot x'_{l+1}. \tag{12}$$

We found that the stop gradient operator $\mathrm{sg}(\cdot)$ is crucial for preserving gradients. The gradient norm of GPAS without $\mathrm{sg}(\cdot)$ looks very similar to that of Pre-LN in Figure 5a. Although it still manages to scale down the activation variance by around 50% across all layers, GPAS without $\mathrm{sg}(\cdot)$ does not offer meaningful improvement in perplexity compared to Pre-LN (see Table 5). This highlights the importance of the stop gradient operator to prevent gradient vanishing issue associated with gradient downscaling.

Learnable vs. predefined gate value. GPAS adopts learnable gates by default. We investigate whether predefined gates can also be used, which could potentially offer more predictability and consistency. To determine a reasonable set of predefined gate values, we extract the gates from

Table 5: Ablations on stop-gradient usage (left) and gating strategy (right) in GPAS.

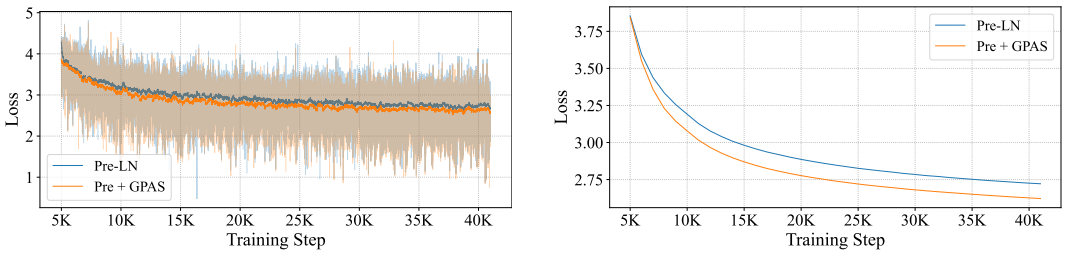
Method	Perplexity
w/ $\mathrm{sg}(\cdot)$	20.35 (-1.00)
w/o $\mathrm{sg}(\cdot)$	21.34 (-0.01)
No GPAS	21.35

Method	Perplexity
Learnable gate	20.35 (-1.00)
Predefined gate	22.46 (+1.11)
No GPAS	21.35

pretrained Pre + GPAS model, and use those values to initialize the gates for the control experiment, where the gate values will be fixed during training. Results are shown in Table 5. We found that fixing the gates in this way led to a substantial drop in performance, especially in early stages when GPAS with learnable gates converges much quicker. This result confirms the importance of learnable gates to account for training dynamics. Another potential way of using predefined gate value is to introduce a warmup stage for the gates, which we leave to future work.

B Pretrain Results on 7B-Parameter Models

To further verify the effectiveness of GPAS on larger scale models, we perform pretraining experiments on Pre-LN and Pre + GPAS with 7B parameters. We follow [41] and use a learning rate of 3×10^{-4} with 10K warmup steps and cosine decay. Batch size is set to 2048 and scheduled to train for 150K steps on 60B tokens. We use gradient clipping of 0.01 on gate parameters α_l and 1.0 on other parameters to stabilize training. Due to constraints on computational resources, we only train the models up to 40K steps with 16B tokens. After 40K training steps, Pre-LN and Pre + GPAS reached evaluation perplexity of 15.27 and 13.82, respectively. As shown in Figure 8b, Pre + GPAS achieves a much faster convergence than vanilla Pre-LN.



(a) Pretrain loss

(b) Evaluation loss

Figure 8: Pretrain and evaluation loss of Pre-LN and Pre + GPAS, 7B parameters

C Gradient Preservation with GPAS

Following Section 3.1, the gradient $\partial\mathcal{L}/\partial x_1$ is:

$$\begin{aligned}
\text{naive scaling:}\quad \frac{\partial\mathcal{L}}{\partial x_1} &= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x_{l+1}}{\partial x_l} \\
&= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\left(\frac{\partial x_{l+1}}{\partial x'_{l+1}}\frac{\partial x'_{l+1}}{\partial x_l}\right) \\
&= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x'_{l+1}}{\partial x_l}\cdot\prod_{l=1}^{L-1}\frac{\partial x_{l+1}}{\partial x'_{l+1}} \\
&= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x'_{l+1}}{\partial x_l}\cdot\prod_{l=1}^{L-1}\beta_l, \\
\text{GPAS:}\quad \frac{\partial\mathcal{L}}{\partial x_1} &= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x'_{l+1}}{\partial x_l}\cdot\prod_{l=1}^{L-1}\frac{\partial x_{l+1}}{\partial x'_{l+1}} \\
&= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x'_{l+1}}{\partial x_l}\cdot\prod_{l=1}^{L-1}I \\
&= \frac{\partial\mathcal{L}}{\partial x_L}\prod_{l=1}^{L-1}\frac{\partial x'_{l+1}}{\partial x_l}.
\end{aligned}$$

D Proof of Theorem 1 of GPAS

Proof. By equation (1) and 2, we have the following:

$$\begin{aligned}
y = x_{\ell+1} &= x'_\ell + \mathrm{FFN}(\frac{1}{\sqrt{\ell}}\mathrm{LN}(x'_\ell)), \\
x'_\ell &= x_\ell + \mathrm{Attn}(\frac{1}{\sqrt{\ell}}\mathrm{LN}(x_\ell)).
\end{aligned} \tag{13}$$

$$\text{Pre-LN: } x'_{l+1} = x_l + f(\mathrm{LN}(x_l)), \quad f \in \{\mathrm{Attn}, \mathrm{FFN}\} \tag{14}$$

$$\text{+ GPAS: } x_{l+1} = x'_{l+1} - \mathrm{SiLU}(\alpha_l)\cdot\mathrm{sg}(x'_{l+1}) \tag{15}$$

Following the variance analysis in [13], both the FFN and Attn modules contribute equally to variance accumulation. For simplicity, we have:

$$\sigma^2_{x_{\ell+1}} = \sigma^2_{x_\ell}(1 + \frac{1}{\sigma_{x_\ell}})\cdot(1 - \mathrm{SiLU}(\alpha_l)). \tag{16}$$

The variance with regard to σ_{x_1}:

$$\sigma^2_{x_\ell} = \sigma^2_{x_1}\Theta\Big(\prod_{k=1}^{\ell-1}\Big(1 + \frac{1}{\sigma_{x_k}}\Big)\Big(1 - \mathrm{SiLU}(\alpha_l)\Big)\Big), \tag{17}$$

For the parameter α_ℓ we have

$$1 - \mathrm{SiLU}(\alpha_\ell) = 1 - \frac{\alpha_\ell}{1 + e^{-\alpha_\ell}} = \frac{(1 + e^{-\alpha_\ell}) - \alpha_\ell}{1 + e^{-\alpha_\ell}}. \tag{18}$$

$$\log\sigma^2_{x_\ell} = \log\sigma^2_{x_1} + \sum_{k=1}^{\ell-1}\log\Big(1 + \frac{1}{\sigma_{x_k}}\Big) + \log\Big(1 - \mathrm{SiLU}(\alpha_\ell)\Big) + C, \tag{19}$$

where C is an (unspecified) constant. Using the original definition for SiLU we have

$$\log\Big(1-\mathrm{SiLU}(\alpha_\ell)\Big)=\log\Big(\frac{1+e^{-\alpha_\ell}-\alpha_\ell}{1+e^{-\alpha_\ell}}\Big)=\log\big(1+e^{-\alpha_\ell}-\alpha_\ell\big)-\log\big(1+e^{-\alpha_\ell}\big). \tag{20}$$

Next, denote $L(\alpha_\ell)=\log\Big(\frac{1+e^{-\alpha_\ell}-\alpha_\ell}{1+e^{-\alpha_\ell}}\Big)$, which is a function of α_ℓ. Thus the inequality becomes

$$\log\sigma_{x_\ell}^2\geq\log\sigma_{x_1}^2+\sum_{k=1}^{\ell-1}\left(\frac{1}{\sigma_{x_k}+1}+L(\alpha_\ell)\right)+C. \tag{21}$$

For fixed $\sigma_{x_\ell}^2$, $\log\sigma_{x_\ell}^2\geq L\cdot(C+L(\alpha_\ell))$, if $\alpha_l<0$, there is no lower bound for the variance. So it can reach 0. To establish a upper bound for $\sigma_{x_\ell}^2$

$$\sum_{k=1}^{\ell-1}\log\Big(1+\frac{1}{\sigma_{x_k}}\Big)\leq\sum_{k=1}^{\ell-1}\frac{1}{\sigma_{x_k}}. \tag{22}$$

The SiLU value of this term is less than or equal to zero (since typically $1-\mathrm{SiLU}(\alpha_\ell)\leq 1$) or it can be bounded by an appropriate constant $M(\alpha_\ell)$. For our derivation, we denote

$$\log\Big(1-\mathrm{SiLU}(\alpha_\ell)\Big)\leq M(\alpha_\ell). \tag{23}$$

Putting these bounds together we have

$$\log\sigma_{x_\ell}^2\leq\log\sigma_{x_1}^2+\sum_{k=1}^{\ell-1}\frac{1}{\sigma_{x_k}}+M(\alpha_\ell)+C. \tag{24}$$

Next, we analyze the gradient stability of GPAS. Following Equation (38) in [13], and applying the formalization techniques from [42], we derive the result under the consideration that $\mathrm{stopgrad}$ is used, thereby eliminating any gradient contributions at that point. Consequently, we obtain:

$$UP\left(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\right)=\prod_{l=1}^{L-1}\left(1+\frac{1}{\sigma_{x_\ell}}A+\frac{1}{\sigma_{x_\ell}^2}B\right), \tag{25}$$

Substitute our lower bound in Equation 21 on $\sigma_{x_l}^2$ into the above product. Notice that if

$$\sigma_{x_l}^2\geq\sigma_{x_1}^2\exp\Big\{S_l\Big\}\quad\text{with}\quad S_l=\sum_{k=1}^{l-1}\left(\frac{1}{\sigma_{x_k}+1}+L(\alpha_\ell)\right)+C, \tag{26}$$

Thus, we have the upper bounds on the inverses:

$$\frac{1}{\sigma_{x_l}}\leq\frac{1}{\sigma_{x_1}}\exp\Big\{-\frac{S_l}{2}\Big\}\quad\text{and}\quad\frac{1}{\sigma_{x_l}^2}\leq\frac{1}{\sigma_{x_1}^2}\exp\Big\{-S_l\Big\}. \tag{27}$$

Thus a valid lower bound for the UP norm is

$$\begin{aligned}UP\left(\left\|\frac{\partial y_L}{\partial x_1}\right\|_2\right)\geq\prod_{l=1}^{L-1}\Bigg(1&+\frac{A}{\sigma_{x_1}}\exp\Big\{-\frac{1}{2}\Big[\sum_{k=1}^{l-1}\left(\frac{1}{\sigma_{x_k}+1}+L(\alpha_\ell)\right)+C\Big]\Big\}\\&+\frac{B}{\sigma_{x_1}^2}\exp\Big\{-\Big[\sum_{k=1}^{l-1}\left(\frac{1}{\sigma_{x_k}+1}+L(\alpha_\ell)\right)+C\Big]\Big\}\Bigg).\end{aligned} \tag{28}$$

Similar to above, assuming that for each layer and based on Equation (24), we have:

$$\sigma_{x_\ell}^2 \leq \sigma_{x_1}^2 \exp\Big\{\sum_{k=1}^{\ell-1} \frac{1}{\sigma_{x_k}} + M(\alpha_\ell) + C\Big\}. \tag{29}$$

So we have:

$$\frac{1}{\sigma_{x_\ell}} \geq \frac{1}{\sigma_{x_1}} \exp\Big\{-\frac{1}{2}\Big[\sum_{k=1}^{\ell-1} \frac{1}{\sigma_{x_k}} + M(\alpha_\ell) + C\Big]\Big\}, \tag{30}$$

Hence the UP product is upper bounded by

$$\begin{aligned} UP\Big(\Big\|\frac{\partial y_L}{\partial x_1}\Big\|_2\Big) \leq \prod_{\ell=1}^{L-1} \Bigg(1 + \frac{A}{\sigma_{x_1}} \exp\Big\{-\frac{1}{2}\Big(\sum_{k=1}^{\ell-1} \frac{1}{\sigma_{x_k}} + M(\alpha_\ell) + C\Big)\Big\} \\ + \frac{B}{\sigma_{x_1}^2} \exp\Big\{-\Big(\sum_{k=1}^{\ell-1} \frac{1}{\sigma_{x_k}} + M(\alpha_\ell) + C\Big)\Big\}\Bigg]). \end{aligned} \tag{31}$$

D.1 Upper bound of $\mathrm{UP}(\cdot)$

Assume that the $M(\alpha_l)$ is bounded and will not change so much. Inserting this into (31), thus, in the case of constant σ and layer-independent $M(\alpha)$, we obtain the explicit bound

$$\begin{aligned} UP\Big(\Big\|\frac{\partial y_L}{\partial x_1}\Big\|_2\Big) \leq \exp\Bigg\{\frac{A}{\sigma} \exp\Big\{-\frac{1}{2}(M(\alpha)+C)\Big\} \frac{1-\exp\Big\{-\frac{L-1}{2\sigma}\Big\}}{1-\exp\Big\{-\frac{1}{2\sigma}\Big\}} \\ + \frac{B}{\sigma^2} \exp\Big\{-\Big(M(\alpha)+C\Big)\Big\} \frac{1-\exp\Big\{-\frac{L-1}{\sigma}\Big\}}{1-\exp\Big\{-\frac{1}{\sigma}\Big\}}\Bigg\}. \end{aligned} \tag{32}$$

It is easy to verify:

$$\frac{1-\exp\Big\{-\frac{L-1}{2\sigma}\Big\}}{1-\exp\Big\{-\frac{1}{2\sigma}\Big\}} = O\Big(\min\{L,\sigma\}\Big) \tag{33}$$

and similarly

$$\frac{1-\exp\Big\{-\frac{L-1}{\sigma}\Big\}}{1-\exp\Big\{-\frac{1}{\sigma}\Big\}} = O\Big(\min\{L,\sigma\}\Big). \tag{34}$$

Plug Equations (33) and (34) into the bound. We deduce

$$\begin{aligned} &UP\Big(\Big\|\frac{\partial y_L}{\partial x_1}\Big\|_2\Big) \\ &\leq \exp\Bigg\{\frac{A}{\sigma} \exp\Big\{-\frac{1}{2}(M(\alpha)+C)\Big\} O\big(\min\{L,\sigma\}\big) + \frac{B}{\sigma^2} \exp\Big\{-(M(\alpha)+C)\Big\} O\big(\min\{L,\sigma\}\big)\Bigg\} \\ &= \exp\Bigg\{O\Bigg(\frac{A}{\sigma} \exp\Big\{-\frac{1}{2}(M(\alpha)+C)\Big\} \min\{L,\sigma\}\Bigg) + O\Bigg(\frac{B}{\sigma^2} \exp\Big\{-(M(\alpha)+C)\Big\} \min\{L,\sigma\}\Bigg)\Bigg\}. \\ &= \exp\Bigg\{O\Bigg(\exp\Big\{-\frac{1}{2}(M(\alpha))\Big\} \frac{\min\{L,\sigma\}}{\sigma}\Bigg)\Bigg\} \end{aligned} \tag{35}$$

D.2 Lower bound for $\mathrm{UP}(\cdot)$

Assume that the $L(\alpha_l)$ is bounded and will not change so much. Inserting this into (28), thus, in the case of constant σ and layer-independent $M(\alpha)$, we can obtain the explicit bound.

Assume with $D = \frac{1}{\sigma+1} + L_(\alpha)$, For each term in the product (with index l), define

$$F(l) = 1 + \frac{A}{\sigma}\exp\Big\{-\frac{1}{2}\Big[(l-1)D + C\Big]\Big\} + \frac{B}{\sigma^2}\exp\Big\{-\Big[(l-1)D + C\Big]\Big\}. \tag{36}$$

Taking the logarithm of the whole product, we get

$$\sum_{s=0}^{L-2}\left\{\frac{A}{\sigma}\exp\Big\{-\frac{1}{2}(sD + C)\Big\} + \frac{B}{\sigma^2}\exp\Big\{-\big(sD + C\big)\Big\} + O\Big(\exp\{-sD - C\}\Big)\right\}. \tag{37}$$

Each of the sums over s is a geometric series. For instance,

$$\sum_{s=0}^{L-2}\exp\Big\{-\frac{1}{2}(sD + C)\Big\} = e^{-C/2}\sum_{s=0}^{L-2}\Big(e^{-D/2}\Big)^s = e^{-C/2}\frac{1 - e^{-(L-1)D/2}}{1 - e^{-D/2}}, \tag{38}$$

and similarly for the second term. Notice that when $e^{-D/2} < 1$ the whole sum (even as $L \to \infty$) is bounded by a constant. Exponentiating, we can write:

$$UP\Big(\Big\|\frac{\partial y_L}{\partial x_1}\Big\|_2\Big) \geq \exp\left\{\frac{A}{\sigma}e^{-C/2}\frac{1}{1 - e^{-D/2}} + \frac{B}{\sigma^2}e^{-C}\frac{1}{1 - e^{-D}} + O(1)\right\}. \tag{39}$$

similar to the above process, we can get:

$$UP\Big(\Big\|\frac{\partial y_L}{\partial x_1}\Big\|_2\Big) \geq \exp\left\{O\left(\frac{1}{\sigma}\frac{1}{1 - e^{(-\frac{1}{\sigma+1} + L(\alpha))/2}} + 1\right)\right\}. \tag{40}$$

□

E License

We provide licenses of assets used in our paper, including model, dataset and code.

- C4 dataset: This is the dataset we used in our pretraining experiments. It's released under the Open Data Commons License Attribution family License.
- Commonsense 170K dataset: This is the dataset we used for supervised fine-tuning, released under MIT License.