Spike No More: Stabilizing the Pre-training of Large Language Models

Sho Takase & Shun Kiyono
SB Intuitions
{sho.takase,shun.kiyono}@sbintuitions.co.jp

Sosuke Kobayashi & Jun Suzuki
Tohoku University
sosk@preferred.jp
jun.suzuki@tohoku.ac.jp

Abstract

Loss spikes often occur during pre-training of large language models. The spikes degrade the performance of large language models and sometimes ruin the pre-training. Since the pre-training needs a vast computational budget, we should avoid such spikes. Based on the assumption that the loss spike is caused by the sudden growth of the gradient norm, we explore factors to keep the gradient norm small through an analysis of the spectral norms of the Jacobian matrices for the sub-layers. Our findings suggest that stabilizing the pre-training process requires two conditions: small sub-layers and large shortcut. We conduct various experiments to empirically verify our theoretical analyses. Experimental results demonstrate that methods satisfying the conditions effectively prevent loss spikes during pre-training.

1 Introduction

Large language models (LLMs) have been fundamental assets for various applications (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023). Increasing the number of parameters in (neural) language models and the number of training data usually leads to better LLMs (Kaplan et al., 2020). Consequently, pre-training requires a vast budget, and thus, minimizing the risk of failure of the pre-training is a paramount concern.

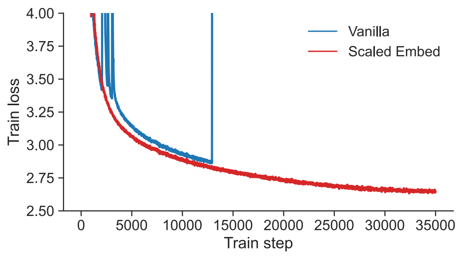


Figure 1: Training loss values of Transformers, whose dimensions and the number of layers are the same as the 1.7 billion parameters configuration in Narayanan et al. (2021). In Vanilla, some spikes occur at the beginning of the training, and its loss value exploded at about 13000 steps.

Despite their widespread use as the foundational architecture for LLMs, a comprehensive theoretical understanding of Transformers (Vaswani et al., 2017) has not yet been achieved. One of the crucial unresolved questions is the reason for the frequent occurrence of pre-training failures in Transformer-based LLMs due to spikes in loss values (loss spike) that can lead to catastrophic divergence (Chowdhery et al., 2022) as illustrated in Vanilla in Figure 1. While several empirical strategies have been proposed to mitigate this problem (Chowdhery et al., 2022; Le Scao et al., 2022; Zeng et al., 2023), the absence of theoretical justification for these methods casts unclear on their generalizability to other situations, such as varying sizes of model parameters.

In this research, we provide theoretical analyses focusing on the loss spike problem during LLM pre-training. We identify the upper bound of the gradient norms for the Transformer-based LLMs through analyses on the spectral norms of the Jacobian matrices for the sub-layers. If the upper bound is large, the gradients may spike suddenly, and we assume that this phenomenon causes the loss spike. Then, we indicate that the upper bound is large in the typical setting, such as the widely used implementation, Megatron-LM (Shoeybi et al., 2020), and thus, the loss spike is likely to occur. In addition, to make the upper bound sufficiently small, we introduce two conditions: (1) initializing the parameters of sub-layers with a small value and (2) making the standard deviation of each embedding close to 1. The former condition can be satisfied by the widely used initialization method for LLMs (Shoeybi et al., 2020; Le Scao et al., 2022; Biderman et al., 2023). On the other hand, the latter condition was satisfied in the original Transformer by scaling embeddings (Vaswani et al., 2017), but such scaling is missing from recent implementations. To sum up, through theoretical analyses, we re-evaluate several previous techniques in terms of the stabilization of LLM pre-training.

Building on our theoretical analysis, we further substantiate our claims through a series of empirical experiments, which provides a clear distinction between effective and ineffective methods over different training scenarios. Our results demonstrate that methods satisfying the conditions avoid the occurrence of loss and gradient spikes. In contrast, methods that fail to meet these conditions remain susceptible to gradient spikes, even when previously recommended as empirical solutions of the loss spike problem. Furthermore, we demonstrate that a method satisfying the conditions enables LLMs to be pre-trained with a comparatively larger learning rate, leading to superior performance outcomes.

2 Preliminary

2.1 Pre-LN Transformer

This paper mainly focuses on the neural architecture used in the GPT series (Radford et al., 2018; 2019; Brown et al., 2020). They use the Pre-LN Transformer (Xiong et al., 2020), which is the de facto standard architecture in recent implementations of Transformers because the training with the architecture is more stable than the original Transformer architecture when we stack many layers (Xiong et al., 2020; Liu et al., 2020; Takase et al., 2023). Let $x \in \mathbb{R}^d$ be an input of a layer of the Transformer, where d denotes the dimension of the layer. The layer outputs y with the following equations:

$$y = x' + \text{FFN}(\text{LN}(x')), \tag{1}$$

$$x' = x + \text{Attn}(\text{LN}(x)), \tag{2}$$

where LN is the layer normalization function[1]. We call the first terms in Equations (1) and (2), i.e., x and x', **shortcut**. In addition, the feed-forward network (FFN) and multi-head self-attention (Attn) are defined as follows[2]:

$$\text{FFN}(x) = W_2(\mathcal{F}(W_1\, x)), \tag{3}$$

$$\text{Attn}(x) = W_O(\text{concat}(\text{head}_1(x), ..., \text{head}_h(x))), \tag{4}$$

$$\text{head}_i(x) = \text{softmax}\left(\frac{(W_{Qi}\, x)^{\mathrm{T}}(W_{Ki}\, X)}{\sqrt{d_{\text{head}}}}\right)(W_{Vi}\, X)^{\mathrm{T}}, \tag{5}$$

where $\mathcal{F}$ is an activation function, concat concatenates input vectors, softmax applies the softmax function to an input vector, and $W_1 \in \mathbb{R}^{d_{\text{ffn}} \times d}$, $W_2 \in \mathbb{R}^{d \times d_{\text{ffn}}}$, $W_{Qi} \in \mathbb{R}^{d_{\text{head}} \times d}$, $W_{Ki} \in \mathbb{R}^{d_{\text{head}} \times d}$, $W_{Vi} \in \mathbb{R}^{d_{\text{head}} \times d}$, 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. In addition, we pack the sequence of input vectors into a matrix as $X \in \mathbb{R}^{d \times L}$, where L is the input sequence length, to compute the self-attention.

[1]We discuss the difference from the architecture using Root Mean Square layer normalization (RMSNorm) (Zhang & Sennrich, 2019) instead of LN in Appendix B.4, and the original Transformer architecture, i.e., Post-LN Transformer in Appendix H.

[2]To simplify equations, we omit bias terms.

2.2 Gradients of Pre-LN Transformers

Let $\mathcal{L}$ be the loss function of the N layered Pre-LN Transformer and J_n be the Jacobian matrix of the n-th layer. We can calculate the gradient of $\mathcal{L}$ using the relations in Equations (1) and (2) as:

$$\frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial \mathcal{L}}{\partial y_N} \prod_{n=1}^{N-1} J_n = \frac{\partial \mathcal{L}}{\partial y_N} \prod_{n=1}^{N-1} \left(\frac{\partial y_n}{\partial x'_n} \frac{\partial x'_n}{\partial x_n} \right), \quad \text{where} \quad J_n = \frac{\partial y_n}{\partial x_n} = \frac{\partial y_n}{\partial x'_n} \frac{\partial x'_n}{\partial x_n}. \tag{6}$$

Using the submultiplicativity of the spectral norm, i.e., $\|AB\|_2 \leq \|A\|_2 \|B\|_2$, and Equation (6), we can derive an upper bound of the norm of the gradient of $\mathcal{L}$ as:

$$\left\| \frac{\partial \mathcal{L}}{\partial x_1} \right\|_2 = \left\| \frac{\partial \mathcal{L}}{\partial y_N} \prod_{n=1}^{N-1} \frac{\partial y_n}{\partial x'_n} \frac{\partial x'_n}{\partial x_n} \right\|_2 \leq \left\| \frac{\partial \mathcal{L}}{\partial y_N} \right\|_2 \prod_{n=1}^{N-1} \left\| \frac{\partial y_n}{\partial x'_n} \right\|_2 \left\| \frac{\partial x'_n}{\partial x_n} \right\|_2. \tag{7}$$

Thus, we can estimate the upper bound of the gradient norm of $\mathcal{L}$ by analyzing the spectral norms of the Jacobian matrices for the FFN layer and the self-attention layer, namely, $\|\frac{\partial y_n}{\partial x'_n}\|_2$ and $\|\frac{\partial x'_n}{\partial x_n}\|_2$.

2.3 Motivation to Suppress the Upper Bound

In our preliminary experiments, when the gradient norms grow suddenly during LLM pre-training, we observe that the loss spike problem is likely to occur. Thus, we assume that we can prevent the loss spike problem by maintaining the gradient norm small. To prevent the growth of the gradient norm, we explore the way to suppress the upper bound described by Equation (7). To suppress the upper bound, we analyze the Jacobian matrices to find a factor to control the upper bound in the following sections, and then, provide two conditions: **small sub-layers** and **large shortcut**. We verify our assumption and theoretical analyses through experiments on LLM pre-training.

3 Analyses on Gradients of Sub-Layers

For the theoretical analyses in this section, we employ the following assumption:

Assumption 1. Let x and x' be the input and intermediate vectors of each layer. Moreover, let W_* denote the model parameter matrix in each layer. We assume that x, x', and W_* for all layers follow a normal distribution with a mean of 0, i.e., $\mu = 0$.

This assumption is valid when we initialize parameters with the normal distribution, the number of heads in Equation (4) is 1, and $\mathcal{F}$ is an identity function. Empirically, the outputs of each sub-layer are close to the normal distribution as illustrated in Appendix D.

3.1 Jacobian Matrix of FFN

Based on Equation (1), $\|\frac{\partial y_n}{\partial x'_n}\|_2$ in Equation (7) can be rewritten as:

$$\left\| \frac{\partial y}{\partial x'} \right\|_2 = \left\| \frac{\partial (x' + \mathrm{FFN}(\mathrm{LN}(x')))}{\partial x'} \right\|_2 = \left\| I + \frac{\partial (\mathrm{FFN}(\mathrm{LN}(x')))}{\partial x'} \right\|_2. \tag{8}$$

We can then derive an upper bound of $\|\frac{\partial y_n}{\partial x'_n}\|_2$ by applying the subadditivity, i.e., $\|A + B\|_2 \leq \|A\|_2 + \|B\|_2$, and submultiplicativity properties of the spectral norm as follows:

$$\left\| \frac{\partial y}{\partial x'} \right\|_2 \leq 1 + \left\| \frac{\partial \mathrm{FFN}(\mathrm{LN}(x'))}{\partial \mathrm{LN}(x')} \right\|_2 \left\| \frac{\partial \mathrm{LN}(x')}{\partial x'} \right\|_2. \tag{9}$$

The right-hand side of this inequality indicates that we can estimate the upper bound of $\|\frac{\partial y}{\partial x'}\|_2$ by separately computing the spectral norms of Jacobian matrices for FFN and LN.

Regarding the FFN part, we assume that the activation function $\mathcal{F}$ is an identity function[3] to simplify the discussion. Under this assumption, the following equation holds:

$$\left\| \frac{\partial \text{FFN}(\text{LN}(x'))}{\partial \text{LN}(x')} \right\|_2 = \|W_2 W_1\|_2. \tag{10}$$

Therefore, we can straightforwardly derive the relation $\|W_2W_1\|_2 \leq \|W_1\|_2\|W_2\|_2$ from the submultiplicativity of the spectral norm. Furthermore, let σ_1 and σ_2 be the standard deviations of W_1 and W_2, respectively. From Assumption 1, the spectral norms of W_1 and W_2 are obtained by their standard deviations and dimensions (Vershynin, 2018), i.e., $\|W_1\|_2 \approx \sigma_1(\sqrt{d} + \sqrt{d_{\text{ffn}}})$ and $\|W_2\|_2 \approx \sigma_2(\sqrt{d} + \sqrt{d_{\text{ffn}}})$. Finally, we can express an upper bound of the spectral norms of the Jacobian matrices for FFN as the following inequality:

$$\left\| \frac{\partial \text{FFN}(\text{LN}(x'))}{\partial \text{LN}(x')} \right\|_2 \leq \sigma_1\sigma_2(\sqrt{d} + \sqrt{d_{\text{ffn}}})^2, \tag{11}$$

where the right-hand side has the relation $\sigma_1\sigma_2(\sqrt{d} + \sqrt{d_{\text{ffn}}})^2 \approx \|W_1\|_2\|W_2\|_2$.

Next, regarding the LN part, the Jacobian matrix of LN can be written as:

$$\frac{\partial \text{LN}(x')}{\partial x'} = \frac{\sqrt{d}}{\|x'\|_2}\left(I - \frac{x'x'^\top}{\|x'\|_2^2}\right) = \frac{\sqrt{d}}{\sigma_{x'}\sqrt{d}}\left(I - \frac{x'x'^\top}{\sigma_{x'}^2 d}\right) = \frac{1}{\sigma_{x'}}\left(I - \frac{zz^\top}{d}\right). \tag{12}$$

The leftmost equation appears in the proof by Xiong et al. (2020). The second equation uses $\|x'\|_2 = \sigma_{x'}\sqrt{d}$, which can be obtained based on Assumption 1. The last equation is derived from the well-known formula of $z = (x' - \mu_{x'})/\sigma_{x'}$, which converts a normal distribution, x', to the standard normal distribution z, where $\mu_{x'} = 0$ in Assumption 1.

We consider the variance (var) of each element in the matrix $zz^\top$. Since $z_i z_i$ follows $\mathcal{X}^2$ with 1 degree of freedom, and $z_i z_j (i \neq j)$ is the multiplication of two independent values following the standard normal distribution, the variances are as follows:

$$\text{var}(z_i z_j) = \begin{cases} 1 & \text{if } i \neq j \\ 2 & \text{otherwise} \end{cases}. \tag{13}$$

Equation (13) indicates that $\frac{zz^\top}{d} \approx 0$ in LLMs due to $d \gg 1$. Therefore, the spectral norm of the Jacobian matrix of LN can be written as:

$$\left\| \frac{\partial \text{LN}(x')}{\partial x'} \right\| = \frac{1}{\sigma_{x'}}, \quad \text{where} \quad \frac{\partial \text{LN}(x')}{\partial x'} = \frac{1}{\sigma_{x'}} I. \tag{14}$$

Finally, Equation (9) can be rewritten by substituting Equations (11) and (14) as:

$$\left\| \frac{\partial y}{\partial x'} \right\|_2 \leq 1 + \frac{\sigma_1\sigma_2}{\sigma_{x'}} C_{\text{ffn}}, \tag{15}$$

where $C_{\text{ffn}} = (\sqrt{d} + \sqrt{d_{\text{ffn}}})^2$ for the simplification.

According to the discussion in Section 2.3 and Equation (15), the standard deviations, σ_1 and σ_2, of W_1 and W_2, respectively, should be sufficiently small, and the standard deviation, $\sigma_{x'}$, of the shortcut, x', should satisfy $\sigma_1\sigma_2 \ll \sigma_{x'}$ in order to keep the upper bound small.

3.2 Jacobian Matrix of Self-Attention

Similar to FFN, we can rewrite $\|\frac{\partial x'}{\partial x}\|_2$ in Equation (7) by using Equation (2) as:

$$\left\| \frac{\partial x'}{\partial x} \right\|_2 = \left\| \frac{\partial (x + \text{Attn}(\text{LN}(x)))}{\partial x} \right\|_2 = \left\| I + \frac{\partial (\text{Attn}(\text{LN}(x)))}{\partial x} \right\|_2. \tag{16}$$

[3]Appendix E discusses the case where we use the ReLU, SiLU, and SwiGLU as the activation functions, which leads to the same conclusion.

We can then derive an upper bound of $\|\frac{\partial x'}{\partial x}\|_2$ by applying the subadditivity and submultiplicativity of the spectral norm, namely:

$$\left\|\frac{\partial x'}{\partial x}\right\|_2 \leq 1 + \left\|\frac{\partial \mathrm{Attn}(\mathrm{LN}(x))}{\partial \mathrm{LN}(x)}\right\|_2 \left\|\frac{\partial \mathrm{LN}(x)}{\partial x}\right\|_2. \tag{17}$$

Therefore, to estimate the upper bound of $\|\frac{\partial x'}{\partial x}\|_2$, we compute the spectral norms of the Jacobian matrices for Attn and LN.

Let $Z(\cdot) = \mathrm{concat}(\mathrm{head}_1(\cdot), ..., \mathrm{head}_h(\cdot)))$ and let J^Z be the Jacobian of the $Z(\cdot)$[4], we can rewrite the spectral norm of the Jacobian matrix of Attn as:

$$\left\|\frac{\partial \mathrm{Attn}(\mathrm{LN}(x))}{\partial \mathrm{LN}(x)}\right\|_2 = \left\|\frac{\partial W_O Z(\mathrm{LN}(x))}{\partial Z(\mathrm{LN}(x))}\frac{\partial Z(\mathrm{LN}(x))}{\partial \mathrm{LN}(x)}\right\|_2 = \|W_O J^Z\|_2. \tag{18}$$

Therefore, we can straightforwardly derive the relation $\|W_O J^Z\|_2 \leq \|W_O\|_2 \|J^Z\|_2$ from the submultiplicativity of the spectral norm.

Let σ_O be the standard deviation of W_O. The relation $\|W_O\|_2 \approx \sigma_O(2\sqrt{d})$ is derived from Assumption 1. We assign this value to Equation (18) and obtain the following inequality:

$$\left\|\frac{\partial \mathrm{Attn}(\mathrm{LN}(x))}{\partial \mathrm{LN}(x)}\right\|_2 \leq \sigma_O(2\sqrt{d})\|J^Z\|_2. \tag{19}$$

Therefore, we can rewrite Equation (17) by substituting Equations (14) and (19) as follows:

$$\left\|\frac{\partial x'}{\partial x}\right\|_2 \leq 1 + \frac{\sigma_O}{\sigma_x} C_{\mathrm{Attn}}, \tag{20}$$

where $C_{\mathrm{Attn}} = (2\sqrt{d})\|J^Z\|_2$ for the simplification.

Thus, similar to the discussion at the end of Section 3.1, the standard deviation, σ_O, of W_O should be small and the standard deviation, σ_x, of the shortcut, x, should satisfy $\sigma_O \ll \sigma_x$ in order to keep the upper bound small.

4 Conditions to Avoid Spikes

Based on the discussions in Section 3, we have to pay attention to values of σ_1, σ_2, σ_O, and the standard deviation of the shortcut to stabilize the pre-training of LLMs. To make σ_1, σ_2, and σ_O small, we have to initialize the corresponding parameters with a small value. Let us consider the actual settings in detail. The widely used initialization method for LLMs (Shoeybi et al., 2020; Le Scao et al., 2022; Biderman et al., 2023), initializes all parameters with a normal distribution $\mathcal{N}(0, \sigma^2)$ where $\sigma = \sqrt{\frac{2}{5d}}$ (Nguyen & Salazar, 2019), and then scales W_2 and W_O to small values based on the number of layers: $\sqrt{\frac{1}{2N}}$ where N is the number of layers[5]. In this situation, σ_1, σ_2, and σ_O are sufficiently small values.

However, in this situation, the standard deviation of the shortcut is also too small. For example, at shallow layers, the standard deviation is close to $\sqrt{\frac{2}{5d}}$ because the embedding matrix is also initialized by $\mathcal{N}(0, \sigma^2)$ where $\sigma = \sqrt{\frac{2}{5d}}$. Therefore, to increase the standard deviation of the shortcut, we make the standard deviation of each embedding

[4]We discuss the detail of J^Z in Appendix G.

[5]Biderman et al. (2023) also scaled W_2 and W_O to small values in the initialization, but they used the strategy introduced by Wang & Komatsuzaki (2021) instead of scaling with $\sqrt{\frac{1}{2N}}$. However, its property is the same essentially because they initialize W_2 and W_O with $\sigma = \frac{2}{N\sqrt{d}}$ which becomes small based on the number of layers.

close to 1[6]. To achieve this, we introduce two kinds of modification: "Scaled Embed" and "Embed LN"[7]. The Scaled Embed scales embeddings with an appropriate value. For example, we multiply embeddings by $\sqrt{d}$, which was used in the original Transformer paper (Vaswani et al., 2017)[8], and then the standard deviations of embeddings become $\sqrt{\frac{2}{5}}$. The Embed LN applies the LN to embeddings. In fact, Le Scao et al. (2022) reported that the Embed LN strategy prevents loss spikes empirically. These two methods are presented as verification examples rather than proposed methods, and alternative approaches could be employed if the conditions are met.

To demonstrate the actual values of the upper bound described in Equation (15), we take the model with 1.7 billion parameters as an example. In addition to the widely used initialization for LLMs (Vanilla) and the above two modifications: Scaled Embed and Embed LN, we compare Xavier Init, which initializes all parameters with the Xavier initialization (Glorot & Bengio, 2010), as the situation where we do not scale W_2 and W_O based on the number of layers. Figure 2 shows the values of Equation (15) for each layer at the beginning of the pre-training. This figure indicates that the methods without suppressing the upper bound, i.e., Xavier Init and Vanilla, rapidly increase the values especially in shallow layers. In contrast, Scaled Embed and Embed LN keep small values. In summary, to make the upper bound of the gradient norms small for the stabilization of the LLM pre-training, we have to satisfy two conditions: (1) **small sub-layers**; initializing the parameters of sub-layers with a small value and (2) **large shortcut**; making the standard deviation of each embedding close to 1.

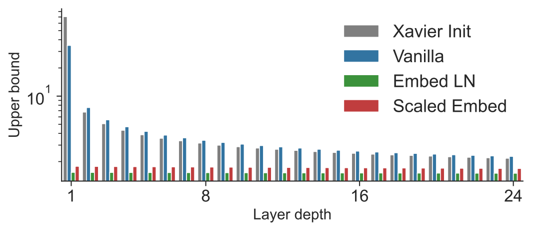


Figure 2: The actual upper bound described in Equation (15) for each Transformer layer at the beginning of the LLM pre-training. Because it is difficult to estimate the strict values for σ_x at all layers, we obtain the empirical values by using some inputs, and assign them to Equation (15).

5 Experiments

We verify the empirical effectiveness of our theoretical analyses. In detail, we demonstrate that controlling the upper bound of the gradient norms also prevents loss and gradient spikes. To assess efficacy in the real situation, we focus on the methods initialized with the widely used method (Shoeybi et al., 2020; Le Scao et al., 2022) in main experiments[9].

5.1 Datasets

We used C4 (Raffel et al., 2020) that consists of clean English texts extracted from Common Crawl as our LLM pre-training corpus. We also used the separated part of C4 as our validation data. We used GPT-2 vocabulary (Radford et al., 2019) that contains Byte Pair Encoding (BPE) subword units (Sennrich et al., 2016) as our vocabulary. To evaluate each method,

[6]Based on Equations (15) and (20), the upper bound becomes small as the standard deviation of the shortcut increases. However, a too large value degrades the performance empirically as described in Appendix C.

[7]We can satisfy the condition by initializing embeddings with the normal distribution $\mathcal{N}(0, \sigma^2)$ where $\sigma = 1$, but we do not adopt this strategy in this study because we use the same initialization method in our experiments.

[8]Although the original Transformer paper introduced this operation, recent implementations ignore this. Therefore, LLMs trained with recent implementations do not satisfy large shortcut.

[9]The Xavier initialization, which does not satisfy the small sub-layers condition, is not widely used for LLMs in recent years. Appendix B.1 shows that the performance of Xavier initialization is worse and fails to avoid spikes.

we computed perplexity on WikiText (Merity et al., 2017) and LAMBADA (Paperno et al., 2016) datasets.

5.2 Model Configurations

As described in Section 2, we used the Pre-LN Transformer architecture. We conduct experiments with two parameter sizes: 350 million (350M) and 1.7 billion (1.7B). We set the learning rate (lr) 5.0×10^{-4}. Appendix A describes more details on the experimental configuration. We compared the following methods. We put ✓ before the method name if the method satisfies both conditions to suppress the upper bound.

Vanilla This is the standard configuration for LLM pre-training. Since this configuration does not suppress the upper bound of the gradient norms, the loss spike is likely to occur.

Embed Detach Zeng et al. (2023) used the shrink embedding gradient technique (Ding et al., 2021) to stabilize their LLM pre-training. This method shrinks gradients on the embedding layer by detaching a part of embeddings from the computational graph as follows:

$$\text{Embed} \leftarrow \gamma\text{Embed} + (1-\gamma)\text{Detach}(\text{Embed}), \tag{21}$$

where γ is a hyper-parameter and Detach detaches an input from the computational graph. We assign 0.1 to γ as in Zeng et al. (2023). Zeng et al. (2023) indicated that this method empirically prevents the loss spike. However, this method does not satisfy the condition on large shortcut, and thus, we show that this method does not completely solve the loss spike.

✓Embed LN Dettmers et al. (2022) and Le Scao et al. (2022) reported that applying the LN to the embedding layer stabilizes their LLM pre-training. As described in Section 4, this method satisfies the conditions to control the upper bound of the gradient norms.

✓Scaled Embed This method multiplies embeddings by $\sqrt{d}$. As described in Section 4, this method satisfies the requirements to control the upper bound of the gradient norms.

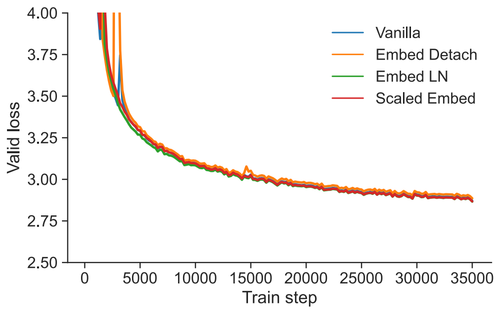
5.3 Main Results

Figure 3 shows the loss values of each method in validation data. Figure 4 shows the gradient norms of each method. These figures indicate that Vanilla and Embed Detach faced loss and gradient spikes. In contrast, Embed LN and Scaled Embed did not face spikes. These results correspond to our theoretical analyses described in Sections 3 and 4. Thus, only methods that make the upper bound of the gradient norm small have successfully avoided spikes in LLM pre-training.

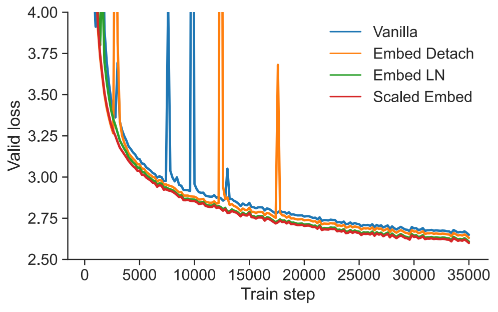
In comparison between 350M and 1.7B parameters, spikes occurred more frequently in 1.7B parameters. Because we initialize embeddings with $\mathcal{N}(0, \sigma^2)$ where $\sigma = \sqrt{\frac{2}{5d}}$, the standard deviations of embeddings become small as d gets larger in Vanilla and Embed Detach. This means that the upper bounds described by Equations (15) and (20) become large as d gets larger because σ_x and σ'_x are nearly equal to the standard deviation of an input embedding in shallow layers. Therefore, if we increase d without any technique to control the upper bound of the gradient norms, a model becomes more unstable. This result corresponds to the previous study reports (Le Scao et al., 2022;

Model	WikiText ↓	LAMBADA ↓
350M parameters		
Vanilla	30.03	24.73
Embed Detach	30.69	26.93
Embed LN	**29.85**	25.03
Scaled Embed	29.86	**24.37**
1.7B parameters		
Vanilla	22.58	15.22
Embed Detach	22.00	13.88
Embed LN	**21.29**	13.00
Scaled Embed	**21.29**	**12.53**

Table 1: Perplexities of each method.

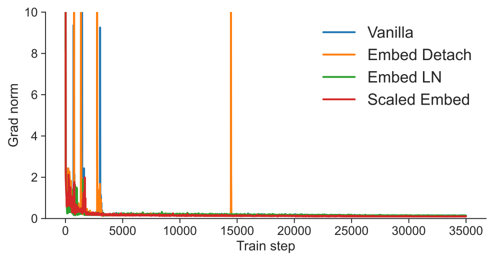


(a) 350M parameters.

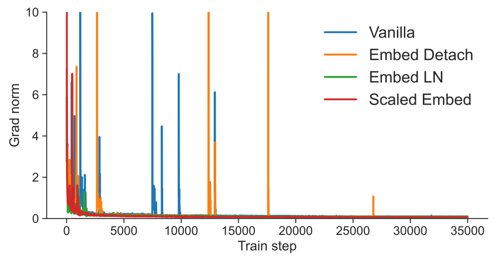


(b) 1.7B parameters.

Figure 3: Loss curves of each method in validation data.



(a) 350M parameters.



(b) 1.7B parameters.

Figure 4: Gradient norms of each method during the training.

Chowdhery et al., 2022; Zeng et al., 2023) that their model became more unstable as they increased the number of parameters.

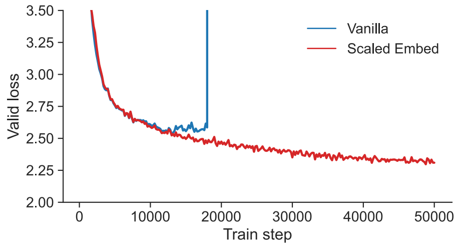
Table 1 shows the perplexities of each method on WikiText and LAMBADA. This table shows that Embed LN and Scaled Embed achieved comparable performance. This result implies that methods have no significant difference from each other in their performance if each method prevents loss and gradient spikes. In contrast, the perplexities of Vanilla and Embed Detach are worse except for Vanilla with 350M parameters in LAMBADA, and the difference in the performance is larger in a large amount of parameters. This result implies that addressing spikes has a more serious influence on the performance as the parameter size gets larger. We conduct experiments on a larger model in the following subsection. Moreover, we discuss other settings in Appendix B.

5.4 Results on 13B Parameter Model

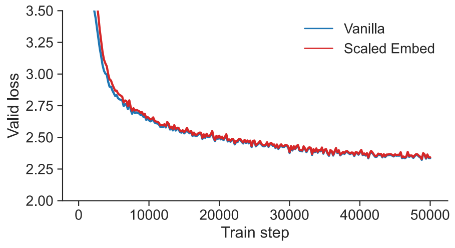
We conduct experiments on pre-trainings of 13B parameter models to indicate that our provided conditions can stabilize a model with many more parameters than the ones discussed in Section 5.3. Due to the limitation of our computational budget, we focused on the comparison between Vanilla and Scaled Embed. We tried two learning rates: 3.0×10^{-4} and 1.0×10^{-4}. Appendix A describes more details about hyper-parameters.

Figure 5 shows the loss values of each configuration in validation data. As shown in (a) of this figure, the loss value of Vanilla rose from approximately 10000 steps in $\text{lr} = 3.0 \times 10^{-4}$. Then, the gradient of this model became too large to continue its pre-training. In contrast, the loss value of Scaled Embed consistently decreased. This result indicates that Scaled Embed stabilized the pre-training of the model with a large number of parameters.

Table 2 shows the perplexities of each configuration in evaluation data. This table indicates that we can achieve better performance when we use a larger learning rate. In addition, the perplexities of Scaled Embed were comparable to ones of Vanilla when we used the small



(a) $\mathrm{lr} = 3.0 \times 10^{-4}$.



(b) $\mathrm{lr} = 1.0 \times 10^{-4}$.

Figure 5: Loss values of each method with 13B parameters when we use two learning rates: $\mathrm{lr} = 3.0 \times 10^{-4}$ and 1.0×10^{-4}

	WikiText ↓		LAMBADA ↓	
Model	$\mathrm{lr} = 3.0 \times 10^{-4}$	$\mathrm{lr} = 1.0 \times 10^{-4}$	$\mathrm{lr} = 3.0 \times 10^{-4}$	$\mathrm{lr} = 1.0 \times 10^{-4}$
Vanilla	N/A	15.12	N/A	6.50
Scaled Embed	**14.47**	15.25	**5.97**	6.53

Table 2: Perplexities of each method with 13B parameters when we use two learning rates: $\mathrm{lr} = 3.0 \times 10^{-4}$ and 1.0×10^{-4}.

Model	PIQA ↑	OpenBookQA ↑	HellaSwag ↑	WinoGrande ↑	ARC-easy ↑	ARC-challenge ↑
$\mathrm{lr} = 3.0 \times 10^{-4}$						
Vanilla	N/A					
Scaled Embed	**78.94**	**39.20**	**71.03**	**63.77**	**60.31**	**35.49**
$\mathrm{lr} = 1.0 \times 10^{-4}$						
Vanilla	77.80	38.40	69.10	61.64	57.95	33.53
Scaled Embed	78.07	**39.20**	68.92	62.59	58.92	33.19

Table 3: Performance of each method with 13B parameters on standard tasks.

learning rate: $\mathrm{lr} = 1.0 \times 10^{-4}$. Table 3 shows the performance of each configuration on the standard benchmark datasets: PIQA (Bisk et al., 2020), OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021) and ARC easy and challenge (Clark et al., 2018). This table shows results consistent with the evaluation in terms of perplexity. These results imply that our provided conditions have no considerable risk in pre-training. Thus, we have to satisfy large shortcut in addition to small sub-layers to stabilize the pre-trainings of LLMs.

6 Related Work

Stability To stabilize trainings of Transformer-based neural language models, there have been various discussions on the architecture (Xiong et al., 2020; Liu et al., 2020; Takase et al., 2023; Zeng et al., 2023; Zhai et al., 2023), initialization method (Nguyen & Salazar, 2019; Zhang et al., 2019; Huang et al., 2020; Wang et al., 2022), training strategy (Zhang et al., 2022; Li et al., 2022), and loss function (Chowdhery et al., 2022; Wortsman et al., 2023).

Xiong et al. (2020) theoretically analyzed gradient scales of each part in Transformers, and indicated that the Pre-LN Transformer is more stable than the Post-LN Transformer, that is the original Transformer architecture (Vaswani et al., 2017). Since the Pre-LN Transformer is more stable than the Post-LN Transformer theoretically and empirically, recent studies mainly have used the Pre-LN Transformer to construct an LLM. We also assume using the Pre-LN Transformer in the analysis on the training dynamics in this paper.

To stabilize the LLM pre-training, Le Scao et al. (2022) applied the layer normalization to the embedding layer. Zeng et al. (2023) used shrink embedding gradient technique (Ding et al., 2021). Nishida et al. (2024) proposed weight scaling as reparameterization (WeSaR) which uses additional parameters to scale parameters of internal layers. In this study, we theoretically proved that the layer normalization to the embedding layer controls the upper bound of the gradient norms of sub-layers when we use the widely used initialization method for LLMs (Nguyen & Salazar, 2019; Shoeybi et al., 2020), and thus, it stabilizes the pre-training.

For the initialization methods, recent studies have proposed maximal update parameterization (μP) and its variants as approaches to transfer hyper-parameters from small models to larger ones without incurring the cost of hyper-parameter search Yang & Hu (2021); Yang et al. (2024a;b). However, as described in Yang et al. (2024b), such methods may be ineffective when applied to Transformers, as their underlying assumptions do not accurately reflect the actual Transformer architectures. In studies focusing on Transformers, Nguyen & Salazar (2019) proposed a strategy to initialize parameters of Transformers with small values to stabilize their training. Zhang et al. (2019) and Huang et al. (2020) indicated that we can remove layer normalizations in Transformers if we use their proposed initialization methods. Wang et al. (2022) adjusted initial parameter scales based on the number of layers to stabilize the Post-LN Transformer. In this study, we indicated that the widely used initialization method (Shoeybi et al., 2020), which makes parameters small, is necessary to stabilize the LLM pre-training. Moreover, we proved that we can prevent the loss spike problem by making the standard deviation of embeddings close to 1.

Efficiency As shown in Section 5.4 and Appendix B.2, our modification enables the pre-training with a relatively larger learning rate, and can achieve better performance. Thus, this study can be regarded as on the efficiency of LLM pre-training because our modification can construct a better LLM with a given budget. Strubell et al. (2019) and Schwartz et al. (2019) reported that recent neural methods require substantial computational costs, and thus, they argued that we have to explore a cost-efficient approach. Rajbhandari et al. (2020) proposed ZeRO that reduces memory redundancies during the multi GPU training without increasing communication volume. Dao et al. (2022) focused on GPU memory reads/writes, and proposed FlashAttention that accelerates the speed of attention mechanisms in Transformers. To reduce the number of computations in the attention mechanism, Shazeer (2019) proposed the multi-query attention that shares one key and value across all of the attention heads in each layer. Takase & Kiyono (2023) explored several parameter sharing strategies, and indicated that parameter sharing across some layers can achieve comparable performance to the vanilla model with a small number of parameters. Moreover, several studies have explored a better construction way with a limited budget (Izsak et al., 2021; Takase & Kiyono, 2021). We believe that we can take advantage of their findings to make our LLMs more efficient.

7 Conclusion

This paper explored why large language models (LLMs) sometimes experience loss spikes during pre-training. To provide evidence, we specifically focused on the gradients of sub-layers. We introduced an upper bound for the gradient norms through an analysis of the spectral norms of the Jacobian matrices for the sub-layers. We then theoretically identified two conditions for avoiding loss spikes: small sub-layers and large shortcut. To meet these conditions, we show that using the widely adopted initialization method for LLMs can make the sub-layer parameters small, and that embedding scaling or incorporating layer normalization into the embedding layer can make the standard deviation of each embedding close to 1, resulting in large shortcut. Experimental results indicated that methods satisfying these conditions avoid loss spikes. Furthermore, these methods allow for training with a relatively larger learning rate, leading to improved performance. We hope our theoretical analyses and empirical findings will help avoid wasting valuable time and computational budgets during LLM construction.

Ethics Statement

To stabilize the LLM pre-training, this paper provides theoretical analyses on the spectral norms of the Jacobian matrices for sub-layers to estimate the upper bound of the gradient norm of $\mathcal{L}$. This paper focuses on only the stability of LLM pre-training, and thus, we have to address other issues of LLMs such as hallucinations to use the LLM in a real application.

Reproducibility Statement

We do not aim to propose a novel method in this paper, but we mainly focus on theoretical analyses on the spectral norms of the Jacobian matrices to find the factor to stabilize the pre-training of LLMs. We justify our theoretical analyses through experiments with various situations. To activate our modification, we add only several lines to a widely used implementation, i.e., Megatron-LM[10]. Therefore, we believe that it is easy to reproduce our experimental results. Moreover, we did not modify the internal architecture of Transformers. In particular, Scaled Embed only scales embeddings by a constant value, making it convenient in terms of portability across various implementations, such as Transformers in HuggingFace[11] and vLLM[12].

However, because it is difficult to conclude that our provided conditions completely solve the instability during pre-training of LLMs, it is better to combine other techniques to stabilize the pre-training such as an auxiliary loss described by Chowdhery et al. (2022) to make the pre-training more stable in an actual pre-training situation.

References

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 2397—2430, 2023.

Yonatan Bisk, Rowan Zellers, Ronan bras, Jianfeng Gao, and Choi Yejin. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432–7439, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira,

[10]https://github.com/NVIDIA/Megatron-LM
[11]https://huggingface.co/docs/transformers/main/index
[12]https://github.com/vllm-project/vllm

Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via blockwise quantization. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 19822–19835, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 249–256, 2010.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 4475–4483, 2020.

Peter Izsak, Moshe Berchansky, and Omer Levy. How to train BERT with an academic budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 10644–10652, 2021.

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

Teven Le Scao, Thomas Wang, Daniel Hesslow, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. What language model to train if you have one million GPU hours? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 765–782, 2022.

Conglong Li, Minjia Zhang, and Yuxiong He. The stability-efficiency dilemma: Investigating sequence length warmup for training gpt models. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 26736–26750, 2022.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5763, 2020.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*, pp. 7297–7306, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2381–2391, 2018.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–15, 2021.

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

Kosuke Nishida, Kyosuke Nishida, and Kuniko Saito. Initialization of large language models via reparameterization to mitigate loss spikes. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 22699–22714, 2024.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pp. 1–9, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1525–1534, 2016.

Stephan Peitz, Sarthak Garg, Udhay Nallasamy, and Matthias Paulik. Cross+Self-Attention for transformer models, 2019.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pp. 186–191, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020.

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

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *CoRR*, abs/1907.10597, 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1715–1725, 2016.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3645–3650, 2019.

Sho Takase and Shun Kiyono. Rethinking perturbations in encoder-decoders for fast training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 5767–5780, 2021.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pp. 78–90, 2023.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. B2T connection: Serving stability and performance in deep transformers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3078–3095, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pp. 5998–6008. 2017.

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

Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers, 2022.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities, 2023.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 10524–10533, 2020.

Greg Yang and Edward J. Hu. Tensor programs VI: Feature learning in infinite-width neural networks. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning, 2024a. URL https://arxiv.org/abs/2310.17813.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024b.

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

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Josh M. Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

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

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

A Hyper-Parameters

Table 4 shows that hyper-parameters used in our experiments on LLMs. To make the experiments on 13B parameter models close to a realistic situation, we increased the batch size and the number of updates in comparison to ones in experiments on 350M and 1.7B parameter models. In addiiton, for Adam β_2, most studies have used 0.95 to stabilize their pre-trainings (Brown et al., 2020; Zhang et al., 2022; Zeng et al., 2023; Biderman et al., 2023; Touvron et al., 2023), and thus, we also used 0.95 in the experiments on 13B parameter models. As described in Section 5.4, we emphasize that the pre-training of Vanilla is essentially unstable even if we use the widely used Adam β_2 value, 0.95, which is known as the technique to stabilize the pre-training. Therefore, our provided conditions are also effective for the stabilization in the realistic situation.

B Discussions on Other Configurations

In this section, we conduct experiments on other configurations to describe connections with previous study reports.

B.1 Methods without Small Sub-Layers

Since we applied the widely used initialization method for LLMs in experiments in Section 5, all methods satisfy the condition on the small sub-layers. In this section, we empirically investigate the property of the method that violates the condition. We compare the Transformer initialized by the Xavier initialization (Glorot & Bengio, 2010) (Xavier Init), and the combination of Xavier Init and Scaled Embed (Xavier Init + Scaled Embed) with Scaled Embed. As described in Table 5, Xavier Init violates both conditions, and Xavier Init + Scaled Embed satisfies the only large shortcut condition. In the same manner as in Section

Table 4: Hyper-parameters used in our experiments on the LLM pre-training.

Name	350M	1.7B	13B
Layer num	24	24	40
Hidden dim size	1024	2304	5120
FFN dim size	4096	9216	20480
Attention heads	16	24	40
Dropout rate	0.1	0.1	0.1
Precision	`float16`	`float16`	`float16`
Sequence length	2048	2048	2048
Batch size	528	528	1024
The number of updates	35000	35000	50000
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.95
Gradient clipping	1.0	1.0	1.0
lr decay style	cosine	cosine	cosine
lr warmup fraction	0.05	0.05	0.05
Weight decay	0.01	0.01	0.01

Method	Small sub-layers	Large shortcut
Xavier Init	-	-
Vanilla	✓	-
Embed Detach	✓	-
Embed LN	✓	✓
Scaled Embed	✓	✓
Xavier Init + Scaled Embed	-	✓

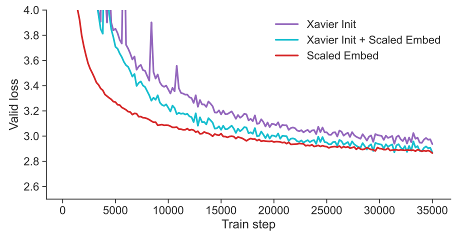
Table 5: Relations between each method in experiments and two conditions to control the upper bound of gradient norms.

Model	WikiText ↓	LAMBADA ↓
350M parameters		
Xavier Init	33.92	34.72
Xavier Init + Scaled Embed	30.50	26.55
Scaled Embed	**29.86**	**24.37**
1.7B parameters		
Xavier Init	30.10	29.29
Xavier Init + Scaled Embed	23.16	15.49
Scaled Embed	**21.29**	**12.53**

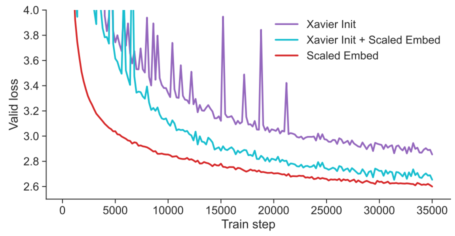
Table 6: Perplexities of each method.

5, we trained models of 350M and 1.7B parameters with $\text{lr} = 5.0 \times 10^{-4}$. We also used the hyper-parameters described in Table 4.

Figure 6 shows loss curves in validation data for 350M and 1.7B parameters in each method, and Figure 7 shows their gradient norms. These figures show that Xavier Init and Xavier Init + Scaled Embed faced loss and gradient spikes. In particular, the spikes appeared more frequently in Xavier Init, which violates both conditions, in comparison with Xavier Init + Scaled Embed. In contrast, Scaled Embed, which satisfies both conditions, avoided the gradient spike and prevented the loss spike problem. These results indicate that we have to satisfy both conditions: small sub-layers and large shortcut to prevent the loss spike problem. Moreover, Table 6 shows perplexities of each configuration in evaluation data. This table indicates that Scaled Embed achieved better performance than Xavier Init and Xavier Init + Scaled Embed that faced some spikes.

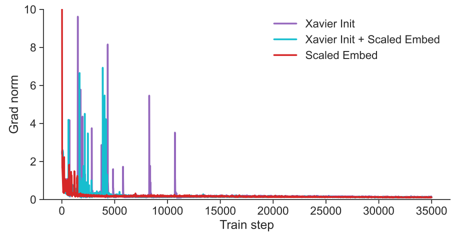


(a) 350M parameters.

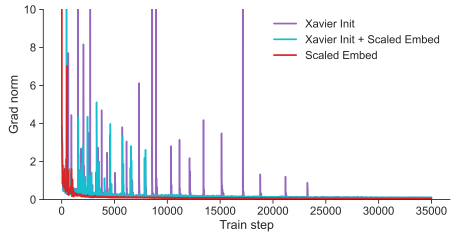


(b) 1.7B parameters.

Figure 6: Loss curves of each method in validation data for the comparison to methods without small sub-layers.

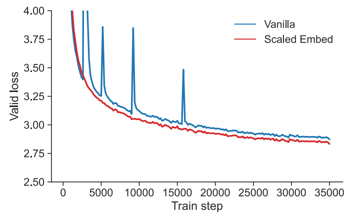


(a) 350M parameters.

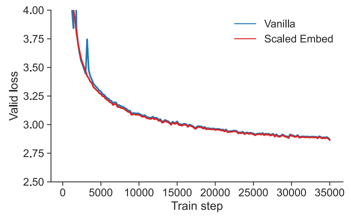


(b) 1.7B parameters.

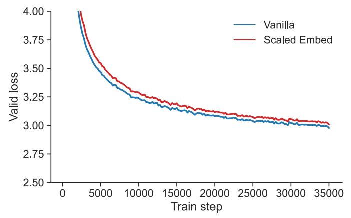
Figure 7: Gradient norms of each method during the training for the comparison to methods without small sub-layers.



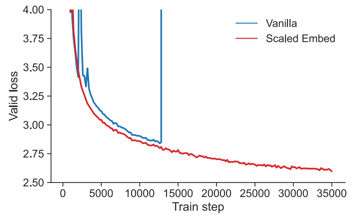
(a) 350M with lr $= 1.0 \times 10^{-3}$.



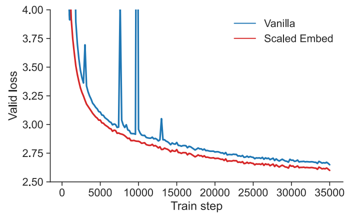
(b) 350M with lr $= 5.0 \times 10^{-4}$.



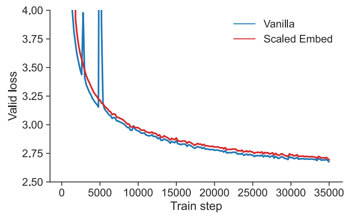
(c) 350M with lr $= 1.0 \times 10^{-4}$.



(d) 1.7B with lr $= 1.0 \times 10^{-3}$.



(e) 1.7B with lr $= 5.0 \times 10^{-4}$.



(f) 1.7B with lr $= 1.0 \times 10^{-4}$.

Figure 8: Loss values of each method with 350M and 1.7B parameters when we vary a learning rate.

B.2 Varying Learning Rate

Le Scao et al. (2022) reported that the stable method, such as Embed LN, was worse than Vanilla. However, in Section 5.3, the stable methods, Scaled Embed and Embed LN, achieved better performance than Vanilla in the 1.7B parameter configuration. We suppose that the difference in the learning rate causes this gap in findings. In this section, we tried to train Vanilla and Scaled Embed with larger and smaller learning rates: lr $= 1.0 \times 10^{-3}$ and 1.0×10^{-4} respectively.

Model	WikiText ↓ lr 1.0×10^{-3}	WikiText ↓ lr 5.0×10^{-4}	WikiText ↓ lr 1.0×10^{-4}	LAMBADA ↓ lr 1.0×10^{-3}	LAMBADA ↓ lr 5.0×10^{-4}	LAMBADA ↓ lr 1.0×10^{-4}
350M parameters						
Vanilla	29.96	30.35	34.51	25.12	24.73	32.49
Scaled Embed	**28.09**	29.86	35.66	**22.03**	24.37	37.14
1.7B parameters						
Vanilla	N/A	22.58	23.54	N/A	15.22	16.17
Scaled Embed	**20.95**	21.29	23.78	**12.26**	12.53	15.39

Table 7: Perplexities of each method with 350M and 1.7B parameters when we vary a learning rate.

Figure 8 shows loss values of each configuration in validation data. As shown in this figure, the larger the learning rate we used, the more frequent the spikes occurred in Vanilla. In particular, in lr $= 1.0 \times 10^{-3}$, the training of Vanilla with 1.7B parameters failed because its gradient exploded. In contrast, Scaled Embed stabilized the training, and thus, its loss values consistently decreased.

Table 7 shows the perplexities of each configuration in evaluation data. This table indicates that Vanilla with 350M parameters achieved better performance in lr $= 1.0 \times 10^{-4}$ that is the situation where its training did not face any spike. This result corresponds to the report of Le Scao et al. (2022). Thus, we suppose that they conducted the comparison with a too-small learning rate to stabilize Vanilla. In contrast, the stable methods are more effective in training with a large learning rate, as shown in Figure 8 and Table 7. Therefore, if Le Scao et al. (2022) used a relatively large learning rate in their experiments, their stable method could achieve better performance.

B.3 Varying Sequence Length

Li et al. (2022) indicated that it is better to train with a short sequence at the early stage to stabilize the LLM pre-training. They justified their method based on the curriculum learning strategy. On the other hand, in this section, we provide the theoretical justification to their method in terms of the standard deviation of the shortcut.

As described in Section 2.1, Transformers add the output of each sub-layer to the shortcut. Since the standard deviation of the self-attention layer tends to decrease with the length of an input sequence especially at the early stage[13], a long sequence tends to keep the standard deviation of the shortcut small. Therefore, the long sequence makes the pre-training of Vanilla more unstable.

We conducted experiments with varying the length of the input sequence L from 128 to 2048. To use the same number of tokens to update parameters, we adjusted the batch size. Figure 9 shows loss values of Vanilla with each L configuration in the validation data. This figure shows that spikes occurred only in the large L, i.e., 1024 and 2048. Moreover, the spikes are more likely to occur at the early stage of the pre-training. Therefore, using a short sequence stabilizes the training at the early stage, as reported in Li et al. (2022).

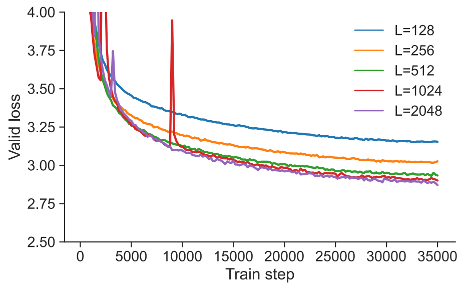
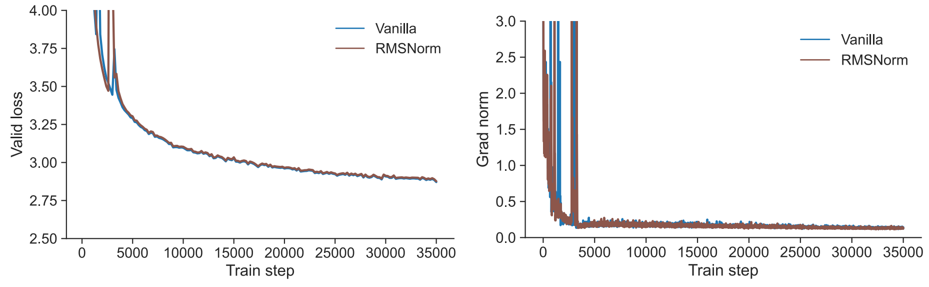


Figure 9: Loss curves of Vanilla with 350M parameters in validation data when we vary the input sequence length. We adjust the batch size to use the same number of tokens for the training of each model.

[13] See Appendix F for details.



(a) Loss curves in validation data. (b) Gradient norms.

Figure 10: Loss values and gradient norms of Vanilla and RMSNorm.

B.4 RMSNorm

Some recent LLMs use the RMSNorm (Zhang & Sennrich, 2019) instead of the LN in their Transformers (Touvron et al., 2023). We discuss such an architecture in this section. In the same as LN discussed in Section 3.1, we can obtain the Jacobian matrix of the RMSNorm with the following equation:

$$\left\| \frac{\partial \text{RMSNorm}(x)}{\partial x} \right\|_2 = \frac{1}{\sigma_x} I \tag{22}$$

Thus, the upper bound of the gradient norm is the same in LN if we use RMSNorm.

Figure 10 shows the loss values and gradient norms of the Vanilla configuration in Section 5.3 and the one using RMSNorms instead of LNs ("RMSNorm" in figures) with 350M parameters. We trained them with $\text{lr} = 5.0 \times 10^{-4}$ as in Section 5.3[14]. As shown in these figures, RMSNorm faced loss and gradient spikes in a similar location to the ones of Vanilla. These empirical results also indicate that the RMSNorms have the same problem as LNs regarding the instability.

C Scaling Embeddings with Larger Value

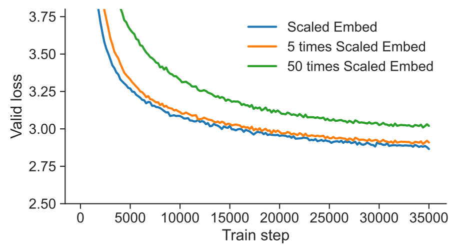
Equations (15) and (20) indicate that we can stabilize the LLM pre-training by adjusting the standard deviation of the shortcut to a large value. In fact, our experimental results show that we can stabilize the LLM pre-training by making the standard deviation of each embedding close to 1. To investigate how about a larger value, we conducted experiments with making the standard deviation of each embedding close to 5 and 50.

Figure 11 shows loss curves in validation data for 350M and 1.7B parameters in each situation. This figure indicates that although all settings prevented the loss spike problem, the larger standard deviation than 1 degraded the performance. Therefore, it is unnecessary to scale the standard deviation of each embedding with a larger value than 1 to prevent the performance degradation.

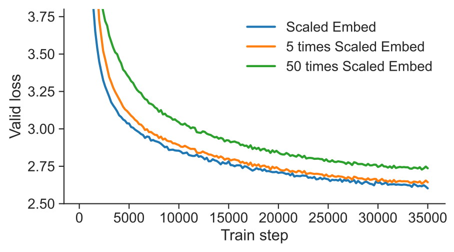
D Distributions of Sub-Layer Outputs

Figures 12 and 13 shows output distributions of each sub-layer at the beginning of pre-training and after pre-training. These figures indicate that each sub-layer output is close to a normal distribution in various configurations both at the beginning of pre-training and after pre-training. Therefore, the assumption in this study, which is that the vector x at each layer follows the normal distribution, is reasonable.

[14]We tried to train them with $\text{lr} = 1.0 \times 10^{-3}$ but RMSNorm exploded.

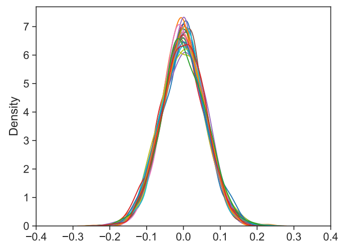


(a) 350M parameters.

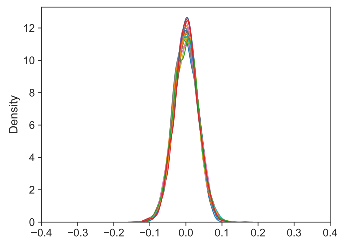


(b) 1.7B parameters.

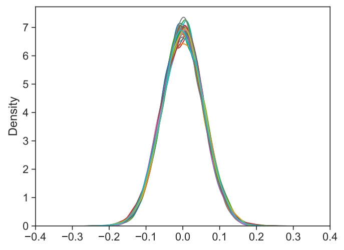
Figure 11: Loss curves in validation data when we scale the standard deviation of embeddings with larger than 1.



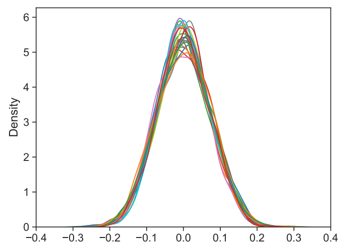
(a) self-attention of 350M parameters.



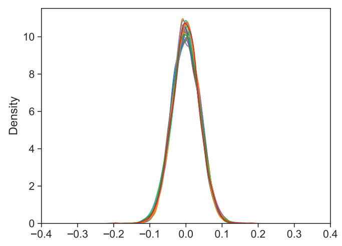
(b) self-attention of 1.7B parameters.



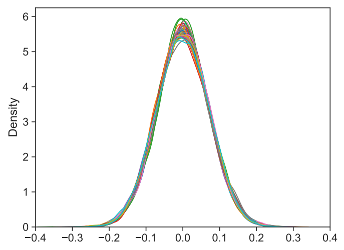
(c) self-attention of 13B parameters.



(d) FFN of 350M parameters.



(e) FFN of 1.7B parameters.



(f) FFN of 13B parameters.

Figure 12: Output distributions of each sub-layer at the beginning of pre-training.

E Discussion on Various Activation Functions in FFN

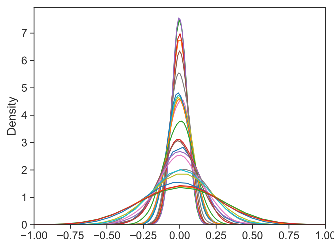
E.1 ReLU

We consider the case where we use the ReLU function as $\mathcal{F}$ instead of the identity function. Because we assume that parameters and the input vector at each layer follow the normal distribution, the internal layer also follows the normal distribution. Therefore, each element of the FFN internal layer is a negative value with half probability. In this case, we can regard that the ReLU function cuts the elements of the internal layer by half. Thus, we replace $W_1 \in \mathbb{R}^{d_{\text{ffn}} \times d}$ and $W_2 \in \mathbb{R}^{d \times d_{\text{ffn}}}$ with $W_1 \in \mathbb{R}^{\frac{d_{\text{ffn}}}{2} \times d}$ and $W_2 \in \mathbb{R}^{d \times \frac{d_{\text{ffn}}}{2}}$ in the discussion in Section 3.1 when we use the ReLU function as $\mathcal{F}$.

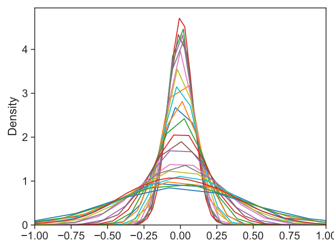
E.2 SiLU

We consider the case where we use the SiLU function as $\mathcal{F}$. The definition of the SiLU function is as follows:

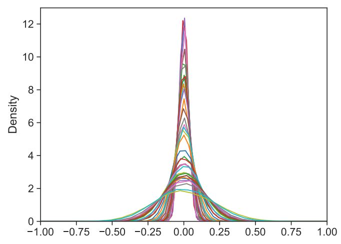
$$\text{SiLU}(x) = x \circ \text{Sigmoid}(x) \tag{23}$$



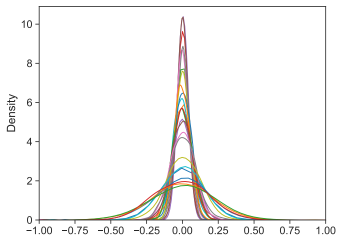
(a) self-attention of 350M parameters.



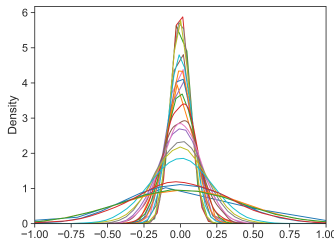
(b) self-attention of 1.7B parameters.



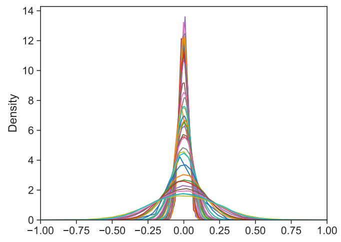
(c) self-attention of 13B parameters.



(d) FFN of 350M parameters.



(e) FFN of 1.7B parameters.



(f) FFN of 13B parameters.

Figure 13: Output distributions of each sub-layer after pre-training.

where Sigmoid is the sigmoid function.

$$\begin{aligned}\frac{\partial \mathrm{SiLU}(x)}{\partial x} &= \mathrm{Sigmoid}(x) + x \circ \frac{\partial \mathrm{Sigmoid}(x)}{\partial x} \\ &= \mathrm{Sigmoid}(x) + x \circ \mathrm{Sigmoid}(x) \circ (1 - \mathrm{Sigmoid}(x)) \\ &= \mathrm{Sigmoid}(x) \circ (1 + x \circ (1 - \mathrm{Sigmoid}(x))) \end{aligned} \tag{24}$$

Let $D = \mathrm{diag}\left(\frac{\partial \mathrm{SiLU}(W_1 x)}{\partial W_1 x}\right) \in \mathbb{R}^{d_{\mathrm{ffn}} \times d_{\mathrm{ffn}}}$. Then, we obtain the Jacobian of the FFN as follows:

$$\frac{\partial \mathrm{FFN}(\mathrm{LN}(x'))}{\partial \mathrm{LN}(x')} = W_2 D W_1 \tag{25}$$

Therefore,

$$\left\| \frac{\partial \mathrm{FFN}(\mathrm{LN}(x'))}{\partial \mathrm{LN}(x')} \right\|_2 \leq \|W_2\|_2 \|D\|_2 \|W_1\|_2 \tag{26}$$

The spectral norm of the diagonal matrix D is the maximum absolute value of its diagonal elements:

$$\|D\|_2 = \max_i \left\| \frac{\partial \mathrm{SiLU}(x_i)}{\partial x_i} \right\| \tag{27}$$

Moreover, we find that its maximum occurs at $x \approx 2.4$ and is approximately 1.1, and thus, $\|D\|_2 \leq 1.1$. Because $\|W_1\|_2 \approx \sigma_1(\sqrt{d} + \sqrt{d_{\mathrm{ffn}}})$ and $\|W_2\|_2 \approx \sigma_2(\sqrt{d} + \sqrt{d_{\mathrm{ffn}}})$, we can express the upper bound as the following inequality:

$$\left\| \frac{\partial \mathrm{FFN}(\mathrm{LN}(x'))}{\partial \mathrm{LN}(x')} \right\|_2 \leq 1.1(\sigma_1 \sigma_2 (\sqrt{d} + \sqrt{d_{\mathrm{ffn}}})^2) \tag{28}$$

Finally, Equation (9) can be rewritten as:

$$\left\| \frac{\partial y}{\partial x'} \right\|_2 \leq 1 + 1.1 \left(\frac{\sigma_1 \sigma_2}{\sigma_{x'}} C_{\mathrm{ffn}} \right) \tag{29}$$

E.3 SwiGLU

We consider the case where we use the SwiGLU function as $\mathcal{F}$. When we use the SwiGLU function, the FFN layer is expressed as follows:

$$\text{FFN}(x) = W_2(\text{Swish}(W_1 x) \circ (Vx)) \tag{30}$$

where $V \in \mathbb{R}^{d_{\text{ffn}} \times d}$, and V follows a normal distribution $\mathcal{N}(0, \sigma_V^2)$. Then, we compute the Jacobian of the FFN(x) as follows:

$$\frac{\partial \text{FFN}(x)}{\partial x} = \frac{\partial \text{FFN}(x)}{\partial W_1 x}\frac{\partial W_1 x}{\partial x} + \frac{\partial \text{FFN}(x)}{\partial V x}\frac{\partial V x}{\partial x} \tag{31}$$

$$\frac{\partial \text{FFN}(x)}{\partial W_1 x} = W_2\left(\text{diag}(Vx) \circ \text{diag}\left(\frac{\partial \text{Swish}(W_1 x)}{\partial W_1 x}\right)\right) \tag{32}$$

$$\frac{\partial \text{FFN}(x)}{\partial V x} = W_2(\text{diag}(\text{Swish}(W_1 x))) \tag{33}$$

Therefore, we can rewrite Equation (31) as follows:

$$\frac{\partial \text{FFN}(x)}{\partial x} = W_2\left(\text{diag}(Vx) \circ \text{diag}\left(\frac{\partial \text{Swish}(\text{W}_1\text{x})}{\partial W_1 x}\right)\right) W_1 + W_2(\text{diag}(\text{Swish}(W_1 x)))V \tag{34}$$

Let $J_1 = W_2\left(\text{diag}(Vx) \circ \text{diag}\left(\frac{\partial \text{Swish}(W_1 x)}{\partial W_1 x}\right)\right) W_1$ and $J_2 = W_2(\text{diag}(\text{Swish}(W_1 x)))V$, $\frac{\partial \text{FFN}(x)}{\partial x} = J_1 + J_2$. We can derive the upper bound of $\|\frac{\partial \text{FFN}(x)}{\partial x}\|_2$ as follows:

$$\left\|\frac{\partial \text{FFN}(x)}{\partial x}\right\|_2 \leq \|J_1\|_2 + \|J_2\|_2 \tag{35}$$

For $\|J_1\|_2$, we have:

$$\|J_1\|_2 \leq \|W_2\|_2 \|\text{diag}(Vx)\|_2 \left\|\text{diag}\left(\frac{\partial \text{Swish}(W_1 x)}{\partial W_1 x}\right)\right\|_2 \|W_1\|_2 \tag{36}$$

Each element of Vx is a sum of d independent random variables with variance $\sigma_x^2 \sigma_V^2$, and thus, $\text{var}(Vx) = d\sigma_x^2\sigma_V^2$. Therefore, from the expected maximum of d_{ffn} Gaussian random variables,

$$\|\text{diag}(Vx)\|_2 \leq \sigma_x \sigma_V \sqrt{2\, d \log d_{\text{ffn}}} \tag{37}$$

The derivation of the Swish function is bounded by 1.1 in the same manner as the SiLU function:

$$\left\|\frac{\partial \text{Swish}(W_1 x)}{\partial W_1 x}\right\|_2 \leq 1.1 \tag{38}$$

The spectral norms of W_1 and W_2 can be obtained $\|W_1\|_2 \approx \sigma_1(\sqrt{d} + \sqrt{d_{\text{ffn}}})$ and $\|W_2\|_2 \approx \sigma_2(\sqrt{d} + \sqrt{d_{\text{ffn}}})$ as described in Section 3.1. We can obtain the upper bound of $\|J_1\|_2$ with these equations:

$$\|J_1\|_2 \leq 1.1 \sigma_x \sigma_V \sigma_1 \sigma_2 (\sqrt{d} + \sqrt{d_{\text{ffn}}})^2 \sqrt{2\, d \log d_{\text{ffn}}} \tag{39}$$

For $\|J_2\|_2$, we have:

$$\|J_2\|_2 \leq \|W_2\|_2 \|(\text{diag}(\text{Swish}(W_1 x)))\|_2 \|V\|_2 \tag{40}$$

Due to $|\text{Swish}(W_1 x)| \leq |W_1 x|$ and $\text{var}(W_1 x) = d\sigma_x^2 \sigma_1^2$, we can obtain:

$$\|(\text{diag}(\text{Swish}(W_1 x)))\|_2 \leq \sigma_x \sigma_1 \sqrt{2\, d \log d_{\text{ffn}}} \tag{41}$$

Therefore,

$$\|J_2\|_2 \leq \sigma_x \sigma_V \sigma_1 \sigma_2 (\sqrt{d} + \sqrt{d_{\text{ffn}}})^2 \sqrt{2\, d \log d_{\text{ffn}}} \tag{42}$$

Based on these equations, we can derive the upper bound as follows:

$$\begin{aligned}\left\| \frac{\partial \text{FFN}(x)}{\partial x} \right\|_2 &\leq \|J_1\|_2 + \|J_2\|_2 \\ &= 2.1\ \sigma_x \sigma_V \sigma_1 \sigma_2 (\sqrt{d} + \sqrt{d_{\text{ffn}}})^2 \sqrt{2\, d \log d_{\text{ffn}}} \\ &= \sigma_x \sigma_V \sigma_1 \sigma_2 C_{\text{swiglu}} \end{aligned} \tag{43}$$

where C_{swiglu} includes 2.1 $(\sqrt{d} + \sqrt{d_{\text{ffn}}})^2 \sqrt{2\, d \log d_{\text{ffn}}}$ for the simplification.

Finally, we consider the actual Transformer layer that includes layer normalization and residual connection:

$$\left\| \frac{\partial y}{\partial x'} \right\|_2 \leq 1 + \frac{\sigma_V \sigma_1 \sigma_2}{\sigma_{x'}} C_{\text{swiglu}} \tag{44}$$

We note that σ_x in Equation (43) is equal to 1 in the actual Transformer layer because we apply the layer normalization to the input of the FFN layer.

F Relation between Input Length and the Standard Deviation of Self-Attention

We explain that the standard deviation of the self-attention layer becomes small as the input length is long. Because we assume that parameters and the input vector at each layer follow the normal distribution, the expectation of each element of $(W_{Qi}\ x)^{\text{T}}(W_{Ki}\ X)$ is 0. Therefore, the expectation after the softmax function is $\frac{1}{L}$ where L is the length of the input sequence. Thus, the long input sequence decreases the standard deviation of the self-attention layer.

To simplify, we consider the case where the number of self-attention heads is 1. In this case, we can obtain the variance of each calculation with the following equation.

$$\text{var}(W_O(x)) = \text{var}(W_O)\text{var}(x)\ d \tag{45}$$

$$\text{var}(W_V(x)) = \text{var}(\text{W}_\text{V})\text{var}(x)\ d \tag{46}$$

where var represents the variance of the matrix/vector. Thus, the variance of the self-attention layer, $\text{var}(\text{Attn}(x))$, is as follows:

$$\text{var}(\text{Attn}(x)) = \text{var}(W_O)\ d \sum^{L} \frac{\text{var}(\text{W}_\text{V})\text{var}(x)\ d}{L^2} \tag{47}$$

$$= \frac{\text{var}(W_O)\text{var}(\text{W}_\text{V})\text{var}(x) d^2}{L} \tag{48}$$

G Details on Jacobian Matrix of Self-Attention

We can represent $\text{concat}(\text{head}_1(x), ..., \text{head}_h(x))$ with the summation of the matrix multiplications as follows:

$$\text{concat}(\text{head}_1(x), ..., \text{head}_h(x)) = \sum_{i=1}^{h} \text{head}_i W_i \tag{49}$$

where $W_i \in \mathbb{R}^{d_{\text{head}} \times d}$ whose corresponding element is 1 and the others are 0. Let J^i be the Jacobian of the $\text{head}_i(x)$, we can represent J^Z in Section 3.2 as follows:

$$J^Z = \sum_{i=1}^{h} J^i W_i \tag{50}$$

In addition, the self-attention consists of the interaction among inputs and outputs of each position in the sequence. Thus, we add indices to Jacobians to represent the positions of the input and output. Let x_j be the input of the position j, and z_k^i be the i-th head of the output position k, and $J_{kj}^i = \frac{\partial z_k^i}{\partial x_j}$. Because J^Z can be regarded as the Jacobian of the input of the position j, we can convert Equation (51) into the following Equation:

$$J^Z = \sum_{k=1}^{L} \sum_{i=1}^{h} J_{kj}^i W_i \tag{51}$$

where L is the length of the input and output sequences. Therefore, we compute J_{kj}^i to obtain J^Z in Section 3.2.

We can obtain a head of the output position k, i.e., z_k, as follows[15]

$$z_k = \sum_{l=1}^{L} A_{kl} v_l \tag{52}$$

where A_{kl} is the l-th element of the attention vector, softmax $\left(\frac{(W_Q\ x_k)^{\mathrm{T}}(W_K\ X)}{\sqrt{d_{\text{head}}}}\right)$ and v_l is $W_V x_l$. Therefore, to obtain J_{kj}, we differentiate z_k with respect to x_j as:

$$J_{kj} = \frac{\partial z_k}{\partial x_j} = \sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\mathrm{T}} + A_{kl} \frac{\partial v_l}{\partial x_j} \right) \tag{53}$$

$$\frac{\partial v_l}{\partial x_j} = W_V \delta_{lj} \tag{54}$$

$$\delta_{lj} = \begin{cases} 1 & \text{if } l = j \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

Thus,

$$\frac{\partial z_k}{\partial x_j} = \sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\mathrm{T}} \right) + A_{kj} W_V \tag{56}$$

Here, we assume that the attention vector is uniform. In this assumption, since $A_{kj} = \frac{1}{L}$, we can obtain the spectral norm of the second term for Equation (56) as $\|A_{kj} W_V\|_2 \approx \frac{\sigma_V}{L}(\sqrt{d} + \sqrt{d_{\text{head}}})$, where σ_V is the standard deviation of W_V. To calculate the first term, we compute $\frac{\partial A_{kl}}{\partial x_j}$.

$$\frac{\partial A_{kl}}{\partial x_j} = A_{kl} \left(\frac{\partial S_{kl}}{\partial x_j} - \sum_{m=1}^{L} A_{km} \frac{\partial S_{km}}{\partial x_j} \right) \tag{57}$$

$$\frac{\partial S_{kl}}{\partial x_j} = \frac{1}{\sqrt{d_{\text{head}}}} \left(W_Q^{\mathrm{T}} W_K x_l \delta_{kj} + W_K^{\mathrm{T}} W_Q x_k \delta_{lj} \right) \tag{58}$$

Let D_l be $W_Q^{\mathrm{T}} W_K x_l$ and E_k be $W_K^{\mathrm{T}} W_Q x_k$. Then,

$$\frac{\partial S_{kl}}{\partial x_j} = \frac{1}{\sqrt{d_{\text{head}}}} \left(D_l \delta_{kj} + E_k \delta_{lj} \right) \tag{59}$$

[15]To simplify the equations, we omit the index i to represent i-th head from the head and parameters.

Therefore,

$$\frac{\partial A_{kl}}{\partial x_j} = \frac{A_{kl}}{\sqrt{d_{\text{head}}}} \left((D_l \delta_{kj} + E_k \delta_{lj}) - \sum_{m=1}^{L} A_{km}(D_m \delta_{kj} + E_k \delta_{mj}) \right) \tag{60}$$

$$= \frac{A_{kl}}{\sqrt{d_{\text{head}}}} \left(\delta_{lj} E_k - A_{kj} E_k + \delta_{kj} \left(D_l - \sum_{m=1}^{L} A_{km} D_m \right) \right) \tag{61}$$

We assign Equation (61) to the first term of Equation (56) and use the assumption $A_{kj} = \frac{1}{L}$:

$$\sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\text{T}} \right) = \sum_{l=1}^{L} \left(\frac{A_{kl}}{\sqrt{d_{\text{head}}}} \left(\delta_{lj} E_k - A_{kj} E_k + \delta_{kj} \left(D_l - \sum_{m=1}^{L} A_{km} D_m \right) {v_l}^{\text{T}} \right) \right) \tag{62}$$

$$= \frac{1}{L\sqrt{d_{\text{head}}}} \sum_{l=1}^{L} \left(\delta_{lj} E_k - \frac{1}{L} E_k + \delta_{kj} \left(D_l - \sum_{m=1}^{L} \frac{1}{L} D_m \right) {v_l}^{\text{T}} \right) \tag{63}$$

$$= \frac{1}{L\sqrt{d_{\text{head}}}} \left(\sum_{l=1}^{L} \left(\left(\delta_{lj} E_k + \delta_{kj} D_l \right) {v_l}^{\text{T}} \right) + \left(\sum_{l=1}^{L} \left(-\frac{1}{L} E_k - \sum_{m=1}^{L} \frac{1}{L} D_m \right) {v_l}^{\text{T}} \right) \right) \tag{64}$$

$$= \frac{1}{L\sqrt{d_{\text{head}}}} \left(E_k {v_j}^{\text{T}} + \sum_{l=1}^{L} \left(\delta_{kj} D_l {v_l}^{\text{T}} \right) - \left(E_k + \sum_{m=1}^{L} D_m \right) {v_l}^{\text{T}} \right) \tag{65}$$

$$= \frac{1}{L\sqrt{d_{\text{head}}}} \left(E_k {v_j}^{\text{T}} - E_k {v_l}^{\text{T}} + \delta_{kj} \sum_{l=1}^{L} \left(D_l {v_l}^{\text{T}} \right) - \left(\sum_{m=1}^{L} D_m \right) {v_l}^{\text{T}} \right) \tag{66}$$

Based on the assumption that parameters and the vector at each layer follow the normal distribution, we assume that the mean of D_l, E_k, and v_l is 0, and thus, we can obtain their norms from their variances. In addition, we assume that the standard deviation of W_Q, W_K and W_V is σ. Then, $\text{var}(D_l) = \text{var}(E_k) = d d_{\text{head}} \sigma^4$, $\text{var}(\sum_l^L D_l) = L d d_{\text{head}} \sigma^4$, and $\text{var}(v_l) = d\sigma^2$. Thus, $\|E_k v_j^{\text{T}}\|_2 \approx \|E_k v_l^{\text{T}}\|_2 \approx \sigma^3 \sqrt{d^3 d_{\text{head}}^2}$, $\|\sum_{l=1}^{L}(D_l v_l^{\text{T}})\|_2 \approx \|(\sum_{m=1}^{L} D_m) v_l^{\text{T}}\|_2 \approx \sigma^3 \sqrt{L d^3 d_{\text{head}}^2}$. Therefore,

$$\left\| \sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\text{T}} \right) \right\|_2$$

$$\leq \frac{1}{L\sqrt{d_{\text{head}}}} \left(\left\| E_k {v_j}^{\text{T}} \right\|_2 + \left\| E_k {v_l}^{\text{T}} \right\|_2 + \delta_{kj} \left\| \sum_{l=1}^{L} \left(D_l {v_l}^{\text{T}} \right) \right\|_2 + \left\| \left(\sum_{m=1}^{L} D_m \right) {v_l}^{\text{T}} \right\|_2 \right) \tag{67}$$

$$\approx \frac{1}{L\sqrt{d_{\text{head}}}} \left(2\sigma^3 \sqrt{d^3 d_{\text{head}}^2} + (\delta_{kj} + 1)\sigma^3 \sqrt{L d^3 d_{\text{head}}^2} \right) \tag{68}$$

$$= \frac{1}{\sqrt{L}} \left(\delta_{kj} + 1 + \frac{2}{\sqrt{L}} \right) \sigma^3 \sqrt{d^3 d_{\text{head}}} \tag{69}$$

Based on this Equation, we can compute the upper bound of $\|J_{kj}\|_2$:

$$\|J_{kj}\|_2 = \left\| \sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\text{T}} \right) + A_{kj} W_V \right\|_2$$

$$\leq \left\| \sum_{l=1}^{L} \left(\frac{\partial A_{kl}}{\partial x_j} {v_l}^{\text{T}} \right) \right\|_2 + \|A_{kj} W_V\|_2 \tag{70}$$

$$\leq \frac{1}{\sqrt{L}} \left(\delta_{kj} + 1 + \frac{2}{\sqrt{L}} \right) \sigma^3 \sqrt{d^3 d_{\text{head}}} + \frac{\sigma}{L}(\sqrt{d} + \sqrt{d_{\text{head}}}) \tag{71}$$

Moreover, we can compute the upper bound of $\|J^Z\|_2$ from Equation (51) as follows:

$$\|J^Z\|_2 = \left\| \sum_{k=1}^{L} \sum_{i=1}^{h} J^i_{kj} W_i \right\|_2 \tag{72}$$

$$\leq \sum_{k=1}^{L} \sum_{i=1}^{h} \|J^i_{kj}\|_2 \|W_i\|_2 \tag{73}$$

$$\approx h \left(\frac{1}{\sqrt{L}} \left(1 + L + \frac{2L}{\sqrt{L}} \right) \sigma^3 \sqrt{d^3 d_{\text{head}}} + \frac{L\sigma}{L} (\sqrt{d} + \sqrt{d_{\text{head}}}) \right) \tag{74}$$

$$= h \left(\left(\sqrt{L} + 2 + \frac{1}{\sqrt{L}} \right) \sigma^3 \sqrt{d^3 d_{\text{head}}} + \sigma(\sqrt{d} + \sqrt{d_{\text{head}}}) \right) \tag{75}$$

H Comparison with Post-LN Transformer

As described in Section 2.1, recent studies use the Pre-LN Transformer architecture to construct their LLMs because the architecture is more stable. In contrast, some recent studies reported that the Post-LN Transformer, which is the original architecture, can achieve better performance than the Pre-LN if we address the instability issue in the Post-LN, i.e., the vanishing gradient problem (Liu et al., 2020; Takase et al., 2023; Wang et al., 2022). We discuss whether the Pre-LN Transformer entirely underperforms the Post-LN. We conducted experiments on machine translation experiments because previous studies mainly focused on them.

We followed the experimental settings in Takase et al. (2023). Table 8 shows the details of hyper-parameters. We used the WMT English-to-German training dataset (Vaswani et al., 2017; Ott et al., 2018), and evaluated each model in newstest2010-2016. We used the encoder-decoder architecture proposed by Peitz et al. (2019). To stabilize the Post-LN Transformer, we applied DeepNet (Wang et al., 2022) and B2T connection (Takase et al., 2023). We compared them to Scaled Embed, that is, the Pre-LN Transformer with the stabilizing techniques described in this paper.

Table 9 shows the averaged BLEU scores among newstest2010-2016. For the BLEU score calculation, we used SacreBLEU (Post, 2018) to obtain compatible scores (Marie et al., 2021). The signature of SacreBLEU is `BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.5.0`. As shown in this table, we used two learning rates: lr $= 1.0 \times 10^{-3}$ and 3.0×10^{-3}. For lr $= 1.0 \times 10^{-3}$, DeepNet and B2T connection outperformed Scaled Embed. Thus, the Post-LN Transformer-based methods achieved better performance than the Pre-LN Transformer-based method. This result corresponds to reports in previous studies (Liu et al., 2020; Wang et al., 2022; Takase et al., 2023).

On the other hand, for lr $= 3.0 \times 10^{-3}$, Scaled Embed achieved better performance than the others with lr $= 1.0 \times 10^{-3}$, and the training of the others failed due to the exploding gradients. This result indicates that the Pre-LN Transformer-based method can achieve better performance if we use a large learning rate. Therefore, the Pre-LN Transformer (with the stabilizing techniques) is more stable than the Post-LN Transformer-based method, and thus, it can achieve better performance when we use a large learning rate that is too large to train the Post-LN Transformers.

Table 8: Hyper-parameters used in the comparison with Post-LN Transformer.

Name	Value
Layer num	18
Hidden dim size	512
FFN dim size	2048
Attention heads	8
Dropout rate	0.5
Precision	`float16`
Word dropout rate	0.1
Max tokens	7168
Adam β_1	0.9
Adam β_2	0.98
Gradient clipping	0.1
lr decay style	inverse square root
Warmup step	4000
Weight decay	0

Table 9: Averaged BLEU scores among newstest2010-2016.

Model	2010	2011	2012	2013	2014	2015	2016	Average ↑
lr $= 1.0 \times 10^{-3}$								
DeepNet	**24.65**	22.30	**22.87**	26.51	27.29	29.77	34.87	26.89
B2T connection	24.46	**22.42**	22.85	26.51	**27.46**	29.91	34.65	26.89
Scaled Embed	24.32	22.21	22.40	26.38	26.89	**29.98**	34.53	26.67
lr $= 3.0 \times 10^{-3}$								
DeepNet				N/A				N/A
B2T connection				N/A				N/A
Scaled Embed	24.52	22.23	22.86	**26.54**	27.35	29.90	**35.16**	**26.94**