Understanding the Difficulty of Training Transformers

Liyuan Liu[†‡] **Xiaodong Liu**[‡] **Jianfeng Gao**[‡] **Weizhu Chen**[§] **Jiawei Han**[†]
{ll2, hanj}@illinois.edu , {xiaodl,jfgao,wzchen}@microsoft.com
[†]University of Illinois at Urbana-Champaign
[‡]Microsoft Research
[§] Microsoft Dynamics 365 AI

Abstract

Transformers have proved effective in many NLP tasks. However, their training requires non-trivial efforts regarding designing cutting-edge optimizers and learning rate schedulers carefully (*e.g.*, conventional SGD fails to train Transformers effectively). Our objective here is to understand *what complicates Transformer training* from both empirical and theoretical perspectives. Our analysis reveals that unbalanced gradients are not the root cause of the instability of training. Instead, we identify an *amplification effect* that influences training substantially – for each layer in a multi-layer Transformer model, heavy dependency on its residual branch makes training unstable, since it amplifies small parameter perturbations (*e.g.*, parameter updates) and results in significant disturbances in the model output. Yet we observe that a light dependency limits the model potential and leads to inferior trained models. Inspired by our analysis, we propose *Admin* (**Ad**aptive **m**odel **in**itialization) to stabilize stabilize the early stage's training and unleash its full potential in the late stage. Extensive experiments show that Admin is more stable, converges faster, and leads to better performance[1].

较劣的

Post-LN的特点
- 其中残差分支指的是非skip connection而不是残差

Post-LN发散了，因此不在图中

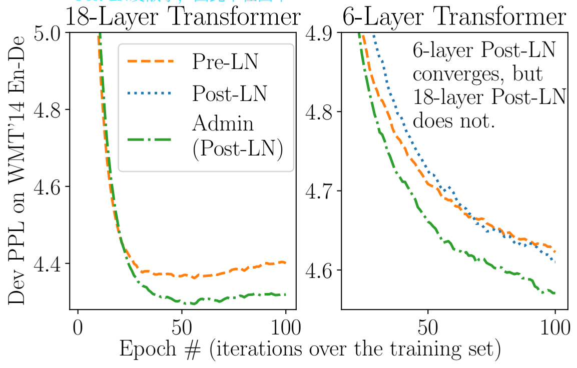


Figure 1: Lacking enough robustness and stability, the 18-Layer Post-LN Transformer training (*i.e.*the original architecture) diverges and is omitted in the left graph. Admin not only stabilizes model training but unleashes the model potential for better performance.

发散

释放, 发挥

1 Introduction

Transformers (Vaswani et al., 2017) have led to a series of breakthroughs in various deep learning tasks (Devlin et al., 2019; Velickovic et al., 2018). They do not contain recurrent connections and can parallelize all computations in the same layer, thus improving effectiveness, efficiency, and scalability. Training Transformers, however, requires extra efforts. For example, although stochastic gradient descent (SGD) is the standard algorithm for conventional RNNs and CNNs, it converges to bad/suspicious local optima for Transformers (Zhang et al., 2019b). Moreover, comparing to other neural architectures, removing the warmup stage in Transformer training results in more severe consequences such as model divergence (Popel and Bojar, 2018; Liu et al., 2020a). Here, we conduct comprehensive analyses in empirical and theoretical manners to answer the question: *what complicates Transformer training*.

Our analysis starts from the observation: the original Transformer (referred to as Post-LN) is less robust than its Pre-LN variant[2] (Baevski and Auli, 2019; Xiong et al., 2019; Nguyen and Salazar, 2019). We recognize that gradient vanishing issue is not the direct reason causing such difference, since fixing this issue alone cannot stabilize Post-LN training. It implies that, besides unbalanced gradients, there exist other factors influencing model training greatly.

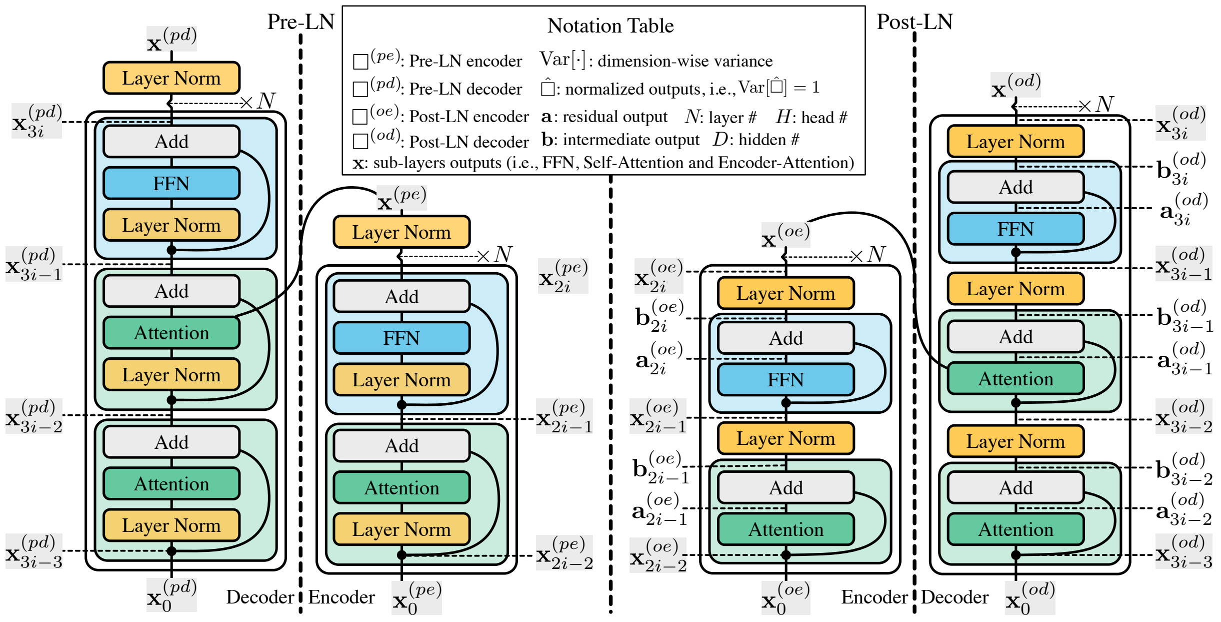
With further analysis, we recognize that for each Transformer residual block, the dependency on its

[1]Implementations are released at: https://github.com/LiyuanLucasLiu/Transforemr-Clinic

[2]As in Figure 2, Post-LN places layer norm outside of residual blocks, and Pre-LN moves them to the inside.

Pre-LN: 在输入Attention或FFN时才进行LN，此时直连特征不进行LN
Post-LN: 在输出Attention或FFN前与直连特征进行LN (TRM中应用)
- 与直连特征相加的称作残差分支，几residual branch

- 奇数层2*i-1为TRM的第i个子层的attention模块
- 偶数层2*i为TRM的第i子层的ffn模块
- x为输出attention模块或ffn模块的输出



Pre-LN
Post-LN
Notation Table
$\square^{(pe)}$: Pre-LN encoder
$\square^{(pd)}$: Pre-LN decoder
$\square^{(oe)}$: Post-LN encoder
$\square^{(od)}$: Post-LN decoder
$\mathrm{Var}[\cdot]$: dimension-wise variance
$\hat{\square}$: normalized outputs, i.e., $\mathrm{Var}[\hat{\square}] = 1$
$\mathbf{a}$: residual output
N: layer #
H: head #
$\mathbf{b}$: intermediate output
D: hidden #
$\mathbf{x}$: sub-layers outputs (i.e., FFN, Self-Attention and Encoder-Attention)
Layer Norm
Add
FFN
Attention
$\times N$
$\mathbf{x}^{(pd)}$
$\mathbf{x}^{(pd)}_{3i}$
$\mathbf{x}^{(pd)}_{3i-1}$
$\mathbf{x}^{(pd)}_{3i-2}$
$\mathbf{x}^{(pd)}_{3i-3}$
$\mathbf{x}^{(pd)}_0$
$\mathbf{x}^{(pe)}$
$\mathbf{x}^{(pe)}_{2i}$
$\mathbf{x}^{(pe)}_{2i-1}$
$\mathbf{x}^{(pe)}_{2i-2}$
$\mathbf{x}^{(pe)}_0$
$\mathbf{x}^{(oe)}$
$\mathbf{x}^{(oe)}_{2i}$
$\mathbf{b}^{(oe)}_{2i}$
$\mathbf{a}^{(oe)}_{2i}$
$\mathbf{x}^{(oe)}_{2i-1}$
$\mathbf{b}^{(oe)}_{2i-1}$
$\mathbf{a}^{(oe)}_{2i-1}$
$\mathbf{x}^{(oe)}_{2i-2}$
$\mathbf{x}^{(oe)}_0$
$\mathbf{x}^{(od)}$
$\mathbf{x}^{(od)}_{3i}$
$\mathbf{b}^{(od)}_{3i}$
$\mathbf{a}^{(od)}_{3i}$
$\mathbf{x}^{(od)}_{3i-1}$
$\mathbf{b}^{(od)}_{3i-1}$
$\mathbf{a}^{(od)}_{3i-1}$
$\mathbf{x}^{(od)}_{3i-2}$
$\mathbf{b}^{(od)}_{3i-2}$
$\mathbf{a}^{(od)}_{3i-2}$
$\mathbf{x}^{(od)}_{3i-3}$
$\mathbf{x}^{(od)}_0$
Decoder
Encoder
Encoder
Decoder

Figure 2: The Architecture and notations of Pre-LN Transformers (Left) and Post-LN Transformers (Right).

residual branch[3] plays an essential role in training stability. First, we find that a Post-LN layer has a heavier dependency on its residual branch than a Pre-LN layer. As in Figure 7, at initialization, a Pre-LN layer has roughly the same dependency on its residual branch and any previous layer, whereas a Post-LN layer has a stronger dependency on its residual branch (more discussions are elaborated in Section 4.1). We find that strong dependencies of Post-LN amplify fluctuations brought by parameter changes and destabilize the training (as in Theorem 2 and Figure 4). Besides, the loose reliance on residual branches in Pre-LN generally limits the algorithm's potential and often produces inferior models.

In light of our analysis, we propose Admin, an adaptive initialization method which retains the merits of Pre-LN stability without hurting the performance. It restricts the layer dependency on its residual branches in the early stage and unleashes the model potential in the late stage. We conduct experiments on IWSLT'14 De-En, WMT'14 En-De, and WMT'14 En-Fr; Admin is more stable, converges faster, and achieves better performance. For example, without introducing any additional hyper-parameters, Admin successfully stabilizes 72-layer Transformer training on WMT'14 En-Fr and achieves a 43.80 BLEU score.

[3]For a residual block $x + f(x)$, its shortcut output refers to x, its residual branch output refers to $f(x)$, and the dependency on its residual branch refers to $\frac{\mathrm{Var}[f(x)]}{\mathrm{Var}[x+f(x)]}$.

2 Preliminaries

Transformer Architectures and Notations. The Transformer architecture contains two types of sub-layers, *i.e.*, Attention sub-layers and Feedforward (FFN) sub-layers. They are composed of mainly three basic modules (Vaswani et al., 2017), *i.e.*, Layer Norm (f_{LN}), Multi-head Attention (f_{ATT}), and Feedforward Network (f_{FFN}).

As illustrated in Figure 2, the Pre-LN Transformer and the Post-LN Transformer organize these modules differently. For example, a Pre-LN encoder organizes the Self-Attention sub-layer as $\mathbf{x}^{(pe)}_{2i-1} = \mathbf{x}^{(pe)}_{2i-2} + f_{\text{S-ATT}}(f_{\text{LN}}(\mathbf{x}^{(pe)}_{2i-2}))$ S-ATT=encoder self-attention and a Post-LN encoder as $\mathbf{x}^{(oe)}_{2i-1} = f_{\text{LN}}(\mathbf{x}^{(oe)}_{2i-2} + f_{\text{S-ATT}}(\mathbf{x}^{(oe)}_{2i-2}))$, where $\mathbf{x}^{(\cdot)}_{2i-2}$ is the input of the i-th Transformer layer and $\mathbf{x}^{(\cdot)}_{2i-1}$ is the output of the i-th Self-Attention sub-layer. Here, we refer $f_{\text{S-ATT}}(f_{\text{LN}}(\mathbf{x}^{(pe)}_{2i-2}))$ and $f_{\text{S-ATT}}(\mathbf{x}^{(oe)}_{2i-2})$ as the residual branches and their outputs as the residual outputs, in contrast to layer/sub-layer outputs, which integrates residual outputs and shortcut outputs.

Notation elaborations are shown in Figure 2. In particular, we use superscripts to indicate network architectures (*i.e.*, the Pre-LN Encoder), use subscripts to indicate layer indexes (top layers have larger indexes), all inputs and outputs are formulated as Sequence-Len × Hidden-Dim.

Layer Norm. Layer norm (Ba et al., 2016) plays a vital role in Transformer architecture. It is defined

Pre-LN，更多地保留恒等映射，深度只作为一个宽度拓展
- encoder，decoder中都会存在越底层梯度越大的情况
Post-LN，更多地保留残差分支，即更关注更深的层
- decoder，外源分布得到的softmax和q相乘，得到的分布方差会有一定的增大
- encoder，同源的分布self-attention, 方差几乎没啥变化

梯度值为子层的输出
(除以了最大值进行归一化)

梯度值为模块的输出
(除以了最大值进行归一化)

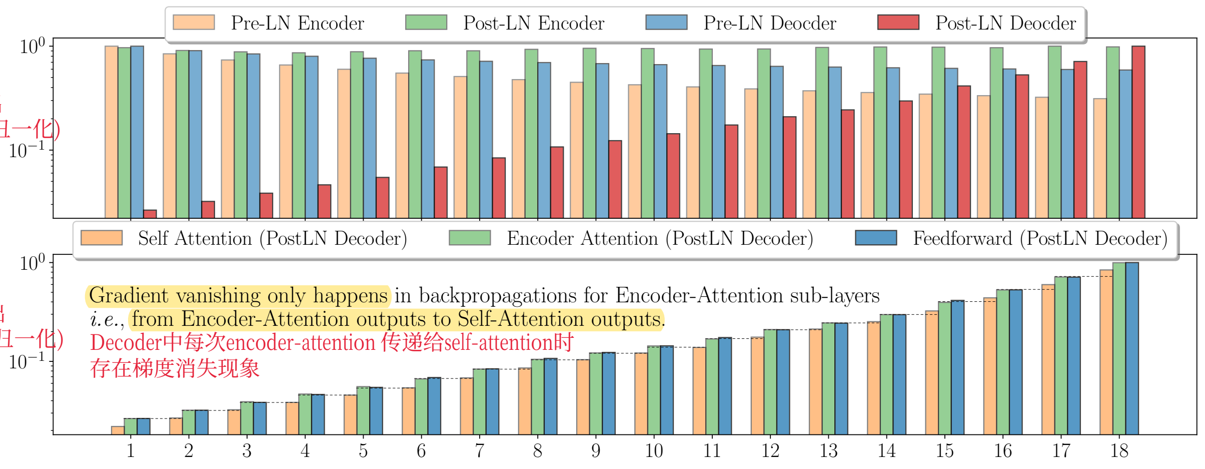


Figure 3: Relative gradient norm histogram (on a log scale) of 18-layer Transformers on the WMT'14 En-De dataset, *i.e.*, the gradient norm of sub-layer outputs, scaled by the largest gradient norm in the same network.

as $f_{\text{LN}}(\mathbf{x}) = \boldsymbol{\gamma}\frac{\mathbf{x}-\mu}{\sigma} + \boldsymbol{\nu}$, where μ and σ are the mean and standard deviation of $\mathbf{x}$.

Feedforward Network. Transformers use two-layer perceptrons as feedforward networks, *i.e.*, $f_{\text{FFN}}(\mathbf{x}) = \phi(\mathbf{x}W^{(1)})W^{(2)}$, where $\phi(\cdot)$ is the non-linear function[4], and $W^{(\cdot)}$ are parameters.

Multi-head Attention. Multi-head Attentions allows the network to have multiple focuses in a single layer and plays a crucial role in many tasks (Chen et al., 2018). It is defined as (with H heads): $f_{\text{ATT}}(\mathbf{q},\mathbf{k},\mathbf{v}) = \sum_{h=1}^{H} f_s(\mathbf{q}W_h^{(Q)}W_h^{(K)}\mathbf{k}^T)\mathbf{v}W_h^{(V_1)}W_h^{(V_2)}$, where f_s is the row-wise softmax function and $W_h^{(\cdot)}$ are parameters. $W_h^{(Q)}$ and $W_h^{(V_1)}$ are $D \times \frac{D}{H}$ matrices, $W_h^{(K)}$ and $W_h^{(V_2)}$ are $\frac{D}{H} \times D$ matrices, where D is the hidden state dimension. Parameters without subscript refer the concatenation of all H-head parameters, *e.g.*, $W^{(Q)} = [W_1^{(Q)}, \cdots, W_H^{(Q)}]$. In Transformer, this module is used in two different settings: Encoder-Attention ($f_{\text{E-ATT}}(\mathbf{x}) = f_{\text{ATT}}(\mathbf{x}, \mathbf{x}^{(\cdot e)}, \mathbf{x}^{(\cdot e)})$ and $\mathbf{x}^{(\cdot e)}$ is the encoder output), and Self-Attention ($f_{\text{S-ATT}}(\mathbf{x}) = f_{\text{ATT}}(\mathbf{x},\mathbf{x},\mathbf{x})$).

3 Unbalanced Gradients

In this study, we strive to answer the question: *what complicates Transformer training*. Our analysis starts from the observation: Pre-LN training is more robust than Post-LN, while Post-LN is more likely to reach a better performance than Pre-LN. In a parameter grid search (as in Figure 10), Pre-LN converges in all 15 settings, and Post-LN diverges in 7 out of 15 settings; when Post-LN converges, it outperforms Pre-LN in 7 out of 8 settings. We seek to reveal the underlying factor that destabilizes Post-LN training and restricts the performance of Pre-LN.

In this section, we focus on the unbalanced gradients (*e.g.*, gradient vanishing). We find that, although Post-LN suffers from gradient vanishing and Pre-LN does not, gradient vanishing is not the direct reason causing the instability of Post-LN. Specifically, we first theoretically and empirically establish that only Post-LN decoders suffer from gradient vanishing and Post-LN encoders do not. We then observe that fixing the gradient vanishing issue alone cannot stabilize training.

3.1 Gradients at Initialization

As gradient vanishing can hamper convergence from the beginning, it has been regarded as the major issue causing unstable training. Also, recent studies show that this issue exists in the Post-LN Transformer, even after using residual connections (Xiong et al., 2019). Below, we establish that only Post-LN decoders suffer from the gradient vanishing, and neither Post-LN encoders, Pre-LN encoders, nor Pre-LN decoders.

We use $\Delta\mathbf{x}$ to denote gradients, *i.e.*, $\Delta\mathbf{x} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ where $\mathcal{L}$ is the training objective. Following previous studies (Glorot and Bengio, 2010), we analyze the gradient distribution at the very beginning of training and find only Encoder-Attention sub-layers in Post-LN suffers from gradient vanishing.

First, we conduct analysis from a theoretical

[4]Our analysis uses ReLU as the activation function, while Admin can be applied to other non-linear functions.

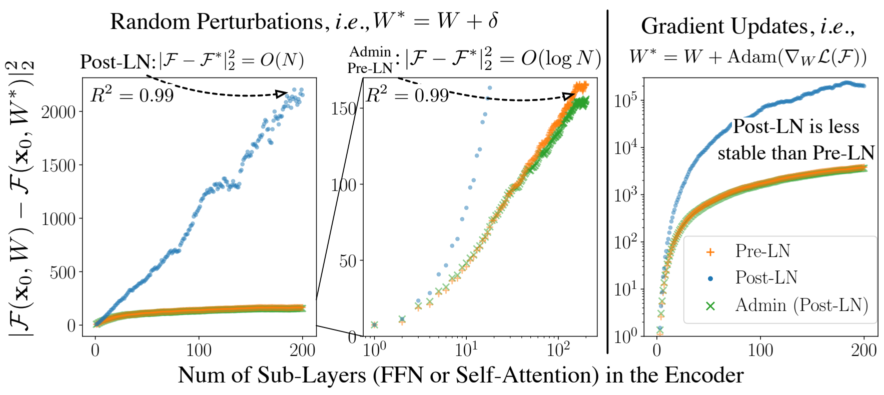


Figure 4: Encoder output changes for parameter changes, *i.e.*, $|\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)|_2^2$ where $W^* - W$ is random perturbations (left) or gradient updates (right). Intuitively, very large $|\mathcal{F} - \mathcal{F}^*|$ indicates the training to be ill-conditioned.

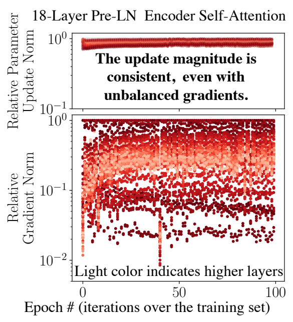


Figure 5: Histogram of relative norm of gradient and $|W_{i+1} - W_i|$ where W_i is the checkpoint saved after training for i epochs.

Encoder	Decoder	Gradient	Training
Post-LN	Post-LN	Varnishing	Diverged
Post-LN	Pre-LN	~~Varnishing~~	Diverged
Pre-LN	Pre-LN	~~Varnishing~~	Converged

Table 1: Changing decoders from Post-LN to Pre-LN fixes gradient vanishing, but does not stabilize model training successfully. Encoder/Decoder have 18 layers.

perspective. Similar to Xiong et al. (2019), we establish that Pre-LN networks do not suffer from gradient vanishing (as elaborated in Appendix A.1). Unlike Xiong et al. (2019), we recognize that not all Post-LN networks suffer from gradient vanishing. As in Theorem 1, we establish that Post-LN Encoder networks do not suffer from gradient vanishing. Detailed derivations are elaborated in Appendix A.2.

THEOREM 1. — For Post-LN Encoders, if γ and $\boldsymbol{\nu}$ in the Layer Norm are initialized as 1 and 0 respectively; all other parameters are initialized by symmetric distributions with zero mean; $\mathbf{x}_i^{(oe)}$ and $\Delta\mathbf{x}_i^{(oe)}$ are subject to symmetric distributions with zero mean; the variance of $\mathbf{x}_i^{(oe)}$ is 1 (*i.e.*, normalized by Layer Norm); $\Delta\mathbf{x}_i^{(oe)}$ and the derivatives of modules in i-th sub-layer are independent, we have $\text{Var}[\Delta\mathbf{x}_{i-1}] \geq \text{Var}[\Delta\mathbf{x}_i]$.

To make sure that the assumptions of Theorem 2 match the real-world situation, we further conduct empirical verification. At initialization, we calculate $||\Delta\mathbf{x}_i^{(\cdot)}||_2$ for 18-layer Transformers[5] and visualize $\frac{||\Delta\mathbf{x}_i^{(\cdot)}||_2}{\max_j ||\Delta\mathbf{x}_j^{(\cdot)}||_2}$ in Figure 3. It verifies that only Post-LN decoders suffer from the gradient vanishing. Besides, we can observe that the dropping of gradient norms mostly happens in the backpropagation from encoder-attention outputs (encoder-attention bars) to its inputs (self-attention bars, since the output of self-attention is the input of encoder-attention). This pattern is further explained in Appendix A.3.

3.2 Impact of the Gradient Vanishing

Now, we explore whether gradient vanishing is the direct cause of training instability.

First, we design a controlled experiment to show the relationship between gradient vanishing and training stability. We construct a hybrid Transformer by combining a Post-LN encoder and a Pre-LN decoder. As in Section 3.1, only Post-LN decoders suffer from gradient vanishing, but not Post-LN encoders. Therefore, this hybrid Transformer does not suffer from gradient vanishing. As shown in Table 1, fixing gradient vanishing alone (*i.e.*, changing Post-LN decoders to Pre-LN decoders) fails to stabilize model training. This observation provides evidence supporting that the gradient vanishing issue is not the direct cause of unstable Post-LN training.

Moreover, we observe that gradients of all attention modules are unbalanced, while adaptive optimizers mostly address this issue. As in Figure 5, adaptive optimizers successfully assign different learning rates to different parameters and lead to consistent update magnitudes even with unbalanced gradients. It explains why the standard SGD fails in training Transformers (*i.e.*, lacking the

[5]Note if $\mathbb{E}[\Delta\mathbf{x}_{i-1}^{(p\cdot)}] = 0$, $\text{Var}[\Delta\mathbf{x}_{i-1}^{(p\cdot)}] \approx |\Delta\mathbf{x}_{i-1}^{(p\cdot)}|_2^2$.

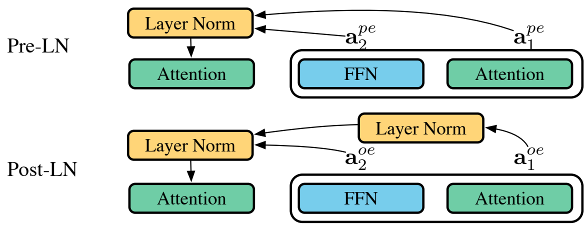


Figure 6: The major difference between Pre-LN and Post-LN is the position of layer norms.

ability to handle unbalanced gradients) and necessitates using adaptive optimizers. More discussions are included in Appendix A.4.

4 Instability from Amplification Effect

We find that unbalanced gradients are not the root cause of the instability of Post-LN, which implies the existence of other factors influencing model training. Now, we go beyond gradient vanishing and introduce the *amplification effect*. Specifically, we first examine the difference between Pre-LN and Post-LN, including their early-stage and late-stage training. Then, we show that Post-LN's training instability is attributed to layer dependency's amplification effect, which intensifies gradient updates and destabilizes training.

4.1 Impact of Layer Norms Positions

As described in Section 2, both Pre-LN and Post-LN employ layer norm to regularize inputs and outputs. Different residual outputs are aggregated and normalized in residual networks before serving as inputs of other layers (*i.e.*, residual outputs will be scaled to ensure the integrated input to have a consistent variance). To some extend, layer norm treats the variance of residual outputs as weights to average them. For example, for Post-LN Self-Attention, we have $\mathbf{x}^{(o\cdot)}_{2i-1} = \frac{\mathbf{x}^{(o\cdot)}_{2i-2}+\mathbf{a}^{(o\cdot)}_{2i-1}}{\sqrt{\mathrm{Var}[\mathbf{x}^{(o\cdot)}_{2i-2}]+\mathrm{Var}[\mathbf{a}^{(o\cdot)}_{2i-1}]}}$ at initialization. Larger $\mathrm{Var}[\mathbf{a}^{(o\cdot)}_{2i-2}]$ not only increases the proportion of $\mathbf{a}^{(o\cdot)}_{2i-2}$ in $\mathbf{x}^{(o\cdot)}_{2i-2}$ but decreases the proportion of other residual outputs. Intuitively, this is similar to the weight mechanism of the weighted average.

The position of layer norms is the major difference between Pre-LN and Post-LN and makes them aggregate residual outputs differently (*i.e.*, using different weights). As in Figure 6, all residual outputs in Pre-LN are only normalized once before feeding into other layers (thus only treating residual output variances as weights); in Post-LN, most

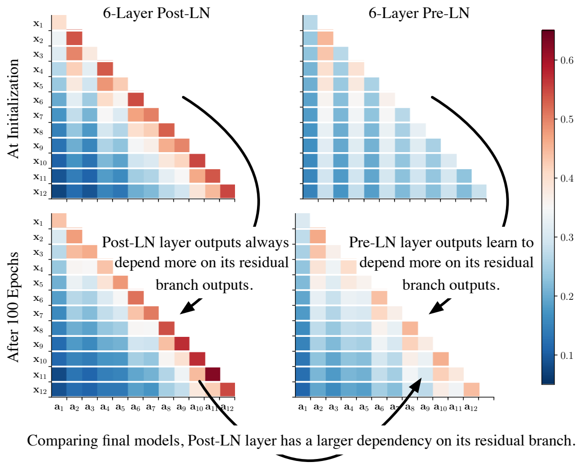


Figure 7: $\beta_{i,j}$ in 6-Layer Post-LN and Pre-LN on the WMT-14 En-De dataset (contains 12 sub-layers).

residual outputs are normalized more than once, and different residual outputs are normalized for different times. For example, if all layers are initialized in the same way, output variances of different Pre-LN residual branches would be similar, and the aggregation would be similar to the simple average. Similarly, for Post-LN, nearby residual outputs are normalized by fewer times than others, thus having relatively larger weights. We proceed to calculate and analyze these weights to understand the impact of layer norm positions.

First, we use $\widehat{\mathbf{a}}_i$ to refer $\frac{\mathbf{a}_i}{\sqrt{\mathrm{Var}\,\mathbf{a}_i}}$ (*i.e.*, normalized outputs of i-th residual branch) and $\widehat{\mathbf{x}}_i$ to refer $\frac{\mathbf{x}_i}{\sqrt{\mathrm{Var}\,\mathbf{x}_i}}$ (*i.e.*, normalized outputs of i-th layer or normalized inputs of (i+1)-th residual branch). Then, we describe their relationships as $\widehat{\mathbf{x}}_i = \sum_{j\le i}\beta_{i,j}\widehat{\mathbf{a}}_j$, where $\beta_{i,j}$ integrates scaling operations of all layer norms (including $\sqrt{\mathrm{Var}[\mathbf{a}_i]}$). For example, Pre-LN sets $\beta_{i,j} = \frac{\sqrt{\mathrm{Var}[\mathbf{a}_j]}}{\sqrt{\mathrm{Var}[\sum_{k\le i}\mathbf{a}_k]}}$. Intuitively, $\beta_{i,j}$ describes the proportion of j-th residual branch outputs in i-th layer outputs, thus reflects the dependency among layers.

We visualize $\beta_{i,j}$ in Figure 7. For a Post-LN layer, its outputs rely more on its residual branch from the initialization to the end. At initialization, Pre-LN layer outputs have roughly the same reliance on all previous residual branches. As the training advances, each layer starts to rely more on its own residual outputs. However, comparing to Post-LN, Pre-LN layer outputs in the final model still has less reliance on their residual branches.

Intuitively, it is harder for Pre-LN layers to depend too much on their own residual branches. In

Pre-LN, layer outputs (*i.e.*, $\mathbf{x}_i^{(p\cdot)}$) are not normalized, and their variances are likely to be larger for higher layers[6]. Since $\beta_{i,i} = \frac{\sqrt{\text{Var}[\mathbf{a}_i]}}{\sqrt{\text{Var}[\mathbf{x}_{i-1}^{(p\cdot)} + \mathbf{a}_i]}}$, $\beta_{i,i}$ is likely to be smaller for higher layers, which restricts i-th layer outputs from depending too much on its residual branch and inhibits the network from reaching its full potential. In other words, Pre-LN restricts the network from being too deep (*i.e.*, if it is hard to distinguish $\mathbf{x}_i^{(p\cdot)}$ and $\mathbf{x}_{i+1}^{(p\cdot)}$, appending one layer would be similar to doubling the width of the last layer), while Post-LN gives the network the choice of being wider or deeper.

4.2 Amplification Effect at Initialization

Although depending more on residual branches allows the model to have a larger potential, it amplifies the fluctuation brought by parameter changes. For a network $\widehat{\mathbf{x}} = \mathcal{F}(\mathbf{x}_0, W)$ where $\mathbf{x}_0$ is the model input and W is the parameter, the output change caused by parameter perturbations is $\text{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)]$, where $W^* = W + \delta$. Its relationship with N is described in Theorem 2, and the derivation is elaborated in Appendix B.

THEOREM 2. — Consider a N-layer Transformer $\widehat{\mathbf{x}} = \mathcal{F}(\widehat{\mathbf{x}}_0, W)$ at initialization, where $\widehat{\mathbf{x}}_0$ is the input and W is the parameter. If the layer dependency stays the same after a parameter change (*i.e.*, $\beta_{i,j}$ has the same value after changing W to W^*, where W is randomly initialized and $\delta = W^* - W$ is independent to W), the output change (*i.e.*, $\text{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)]$) can be estimated as $\sum_{i=1}^{N} \beta_{i,i}^2 C$ where C is a constant.

If $\text{Var}[\mathbf{a}_i]$ is the same for all layers, Pre-LN sets $\beta_{i,i}^2$ as $1/i$, and Post-LN sets $\beta_{i,i}^2$ as a constant. Thus, we have Corollary 1 and 2 as below.

COROLLARY 1. — For a N-layer Pre-LN $\mathcal{F}$, we have $\text{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)] = O(\log N)$.

COROLLARY 2. — For a N-layer Post-LN $\mathcal{F}$, we have $\text{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)] = O(N)$.

They show that, since Post-LN relies more on residual branches than Pre-LN (*i.e.*, has a larger $\beta_{i,i}^2$), the perturbation is amplified to a larger magnitude. To empirically verify these relationships, we calculate $|\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)|_2^2$ for Pre-LN and Post-LN and visualize the results in Figure 4. In Corollary 2, N is linearly associated with $|\mathcal{F} - \mathcal{F}^*|_2^2$ for Post-LN; and in Corollary 1, $\log N$ is linearly associated with $|\mathcal{F} - \mathcal{F}^*|_2^2$ for Pre-LN. These relationships match the observation in our experiments (as in Figure 4). For further verification, we measure their correlation magnitudes by R^2 and find $R^2 = 0.99$ in both cases.

Moreover, we replace the random noise δ with optimization updates (*i.e.*, setting $W^* = W + \text{Adam}(\Delta W)$, where $\text{opt}(\cdot)$ is update calculated by the Adam optimizer) and visualize output shifts. This replacement makes the correlation between $|\mathcal{F} - \mathcal{F}^*|_2^2$ and N (for Post-LN) or $\log N$ (for Pre-LN) to be weaker (*i.e.*, $R^2 = 0.75$). Still, as in Figure 4, the output shift $|\mathcal{F} - \mathcal{F}^*|_2^2$ for Post-LN is larger than Pre-LN by multiple magnitudes.

Intuitively, large output shifts would destabilize the training (Li et al., 2018). Also, as elaborated in Appendix B, the constant C in Theorem 2 is related to network derivatives and would be smaller as training advances, which explains why warmup is also helpful for the standard SGD. Therefore, we conjecture it is the large output shift of Post-LN results in unstable training. We proceed to stabilize Post-LN by controlling the dependency on residual branches in the early stage of training.

4.3 *Admin* – Adaptive Model Initialization

In light of our analysis, we add additional parameters (*i.e.*, $\boldsymbol{\omega}$) to control residual dependencies of Post-LN and stabilize training by adaptively initializing $\boldsymbol{\omega}$ to ensure an $O(\log N)$ output change.

Due to different training configurations and model specificities (*e.g.*, different models may use different activation functions and dropout ratios), it is hard to derive a universal initialization method. Instead, we decompose model initialization into two phrases: *Profiling* and *Initialization*. Specifically, Admin adds new parameters $\boldsymbol{\omega}$ and constructs its i-th sub-layer as $\mathbf{x}_i = f_{\text{LN}}(\mathbf{b}_i)$, where $\mathbf{b}_i = \mathbf{x}_{i-1} \cdot \boldsymbol{\omega}_i + f_i(\mathbf{x}_{i-1})$, $\boldsymbol{\omega}_i$ is a D-dimension vector and $\cdot$ is element-wise product. Then the *Profiling* phrase and *Initialization* phrase are:

Profiling. After initializing the network with a standard method (initializing $\boldsymbol{\omega}_i$ as $\mathbf{1}$), conduct forward propagation without parameter updating and record the output variance of residual branches (*i.e.*, calculate $\text{Var}[f_i(\mathbf{x}_{i-1})]$). Since all elements in the same parameter/output matrix are independent to each other and are subject to the same distribution, it is sufficient to use a small number of instances in

[6] If $\mathbf{a}_0$ and $\mathbf{a}_1$ are independent, $\text{Var}[\mathbf{a}_0 + \mathbf{a}_1] = \text{Var}[\mathbf{a}_0] + \text{Var}[\mathbf{a}_1]$; also, in our experiments $\text{Var}[\mathbf{x}_i^{(p\cdot)}]$ increases as i becomes larger

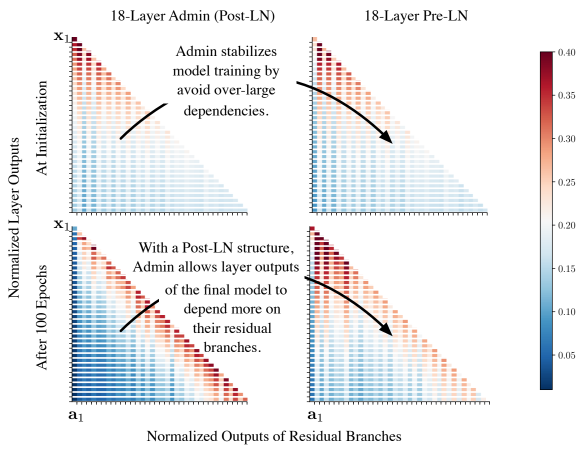


Figure 8: $\beta_{i,j}$ of 18-Layer Admin (Post-LN) and Pre-LN on the WMT-14 En-De dataset.

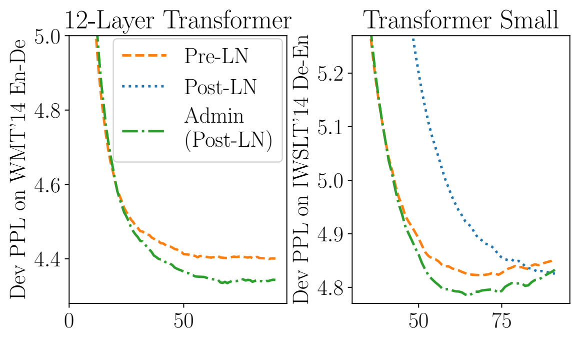


Figure 9: Development PPL on the WMT'14 En-De dataset and the IWLST'14 De-En dataset.

this phrase. In our experiments, the first batch (no more than 8192 tokens) is used.

Initialization. Set $\boldsymbol{\omega}_i = \sqrt{\sum_{j<i} \mathrm{Var}[f_j(\mathbf{x}_{j-1})]}$ and initialize all other parameters with the same method used in the *Profiling* phrase.

In the early stage, Admin sets $\beta_{i,i}^2$ to approximately $\frac{1}{i}$ and ensures an $O(\log N)$ output change, thus stabilizing training. Model training would become more stable in the late stage (the constant C in Theorem 2 is related to parameter gradients), and each layer has the flexibility to adjust $\boldsymbol{\omega}$ and depends more on its residual branch to calculate the layer outputs. After training finishes, Admin can be reparameterized as the conventional Post-LN structure (*i.e.*, removing $\boldsymbol{\omega}$). More implementation details are elaborated in Appendix C.

To verify our intuition, we calculate the layer dependency of 18-Layer models and visualize the result in Figure 8. Figures 7 and 8 show that Admin avoids over-large dependencies at initialization and unleashes the potential to make the layer outputs depend more on their residual outputs in the final model. Moreover, we visualize the output change of Admin in Figure 4. Benefiting from the adaptive initialization, the output change of Admin gets roughly the same increase speed as Pre-LN, even constructed in the Post-LN manner. Also, although Admin is formulated in a Post-LN manner and suffers from gradient vanishing, 18-layer Admin successfully converges and outperforms 18-layer Pre-LN (as in Table 2). This evidence supports our intuition that the large dependency on residual branches amplifies the output fluctuation and destabilizes training.

5 Experiments

We conduct experiments on IWSLT'14 De-En, WMT'14 En-De, and WMT'14 En-Fr. More details are elaborated in Appendix D.

5.1 Performance Comparison

We use BLEU as the evaluation matric and summarize the model performance in Table 2. On the WMT'14 dataset, we use Transformer-base models with 6, 12, or 18 layers. Admin achieves a better performance than Post-LN and Pre-LN in all three settings. Specifically, 12-Layer and 18-Layer Post-LN diverges without the adaptive initialization. Pre-LN converges in all settings, but it results in sub-optimal performance. Admin not only stabilizes the training of deeper models but benefits more from the increased model capacity then Pre-LN, which verifies our intuition that the Pre-LN structure limits the model potential. As in Figure 1 and Figure 9, although the 6-layer Pre-LN converges faster than Post-LN, its final performance is worse than Post-LN. In contrast, Admin not only achieves the same convergence speed with Pre-LN in the early stage but reaches a good performance in the late stage.

We use 6-layer Transformer-small (its hidden dimension is smaller than the base model) on the IWSLT'14 dataset, and all methods perform similarly. Still, as in Figure 10, Admin outperforms the other two by a small margin. Together with WMT'14 results, it implies the training stability is related to layer number. For shallow networks, the stability difference between Post-LN and Pre-LN is not significant (as in Figure 4), and all methods reach reasonable performance. It is worth mentioning that attention and activation dropouts have an enormous impact on IWSLT'14, which is smaller than WMT'14 datasets.

Table 2: BLEU on IWSLT'14 De-En and WMT'14 En-Fr/De (AL-BL refers A-layer encoder & B-layer decoder).

Dataset	IWSLT'14 De-En	WMT'14 En-Fr		WMT'14 En-De		
Enc #–Dec #	6L–6L (small)	6L–6L	60L–12L	6L–6L	12L–12L	18L–18L
Post-LN	35.64±0.23	41.29	failed	27.80	failed	failed
Pre-LN	35.50±0.04	40.74	43.10	27.27	28.26	28.38
Admin	**35.67±0.15**	**41.47**	**43.80**	**27.90**	**28.58**	**29.03**

To further explore the potential of Admin, we train Transformers with a larger size. Specifically, we expand the Transformer-base configuration to have a 60-layer encoder and a 12-layer decoder. As in Table 2, our method achieves a BLEU score of 43.8 on the WMT'14 En-Fr dataset, the new state-of-the-art without using additional annotations (*e.g.*, back-translation). More discussions are conducted in Appendix F to compare this model with the current state of the art. Furthermore, in-depth analyses are summarized in Liu et al. (2020b), including systematic evaluations on the model performance (with TER, METEOR, and BLEU), comprehensive discussions on model dimensions (*i.e.*, depth, head number, and hidden dimension), and fine-grained error analysis. It is worth mentioning that the 60L-12L Admin model achieves a 30.1 BLEU score on WMT'14 En-De (Liu et al., 2020b).

5.2 Connection to Warmup

Our previous work (Liu et al., 2020a) establishes that the need for warmup comes from the unstable adaptive learning rates in the early stage. Still, removing the warmup phrase results in more severe consequences for Transformers than other architectures. Also, warmup has been found to be useful for the vanilla SGD (Xiong et al., 2019).

Theorem 1 establishes that $\mathrm{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)] \approx \sum_{i=1}^{N} \beta_{i,i}^2 C$ where $C = \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}^*_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}^*_{i-1}, W_i^*)]$. In the early stage of training, the network has larger parameter gradients and thus larger C. Therefore, using a small learning rate at initialization helps to alleviate the massive output shift of Post-LN. We further conduct experiments to explore whether more prolonged warmups can make up the stability difference between Post-LN and Pre-LN. We observe that 18-layer Post-LN training still fails after extending the warmup phrase from 8 thousand updates to 16, 24, and 32 thousand. It shows that learning rate warmup alone cannot neutralize the

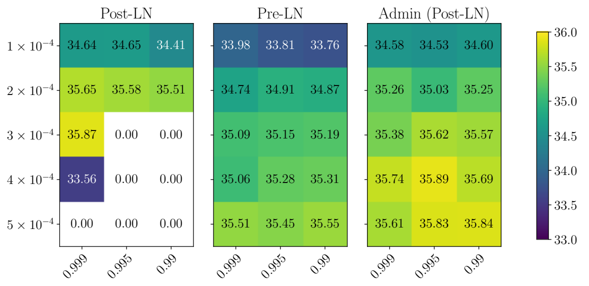


Figure 10: BLEU score of Post-LN, Pre-LN and Admin on the IWSLT'14 De-En dataset (x-axis is the β_2 for adaptive optimizers and y-axis is the learning rate). Pre-LN converges in all settings while Post-LN diverges in 7 out of 15 settings. When Post-LN converges, it outperforms Pre-LN in 7 out of 8 settings. Admin stabilizes Post-LN training and outperforms Pre-LN (its best performance is comparable with Post-LN).

instability of Post-LN. Intuitively, massive output shifts not only require a small learning rate but also unsmoothes the loss surface (Li et al., 2018) and make the training ill-conditioned.

Admin regularizes the model behavior at initialization and stabilizes the training. To explore whether Admin is able to stabilize the training alone, we remove the warmup phase and conduct a grid search on optimizer hyper-parameters. The results are visualized in Figure 10. It shows that as Post-LN is more sensitive to the choice of hyper-parameters, Admin successfully stabilizes the training without hurting its potential.

5.3 Comparing to Other Initializations

We compare our methods with three initialization methods, *i.e.*, ReZero (Bachlechner et al., 2020), FixUp (Zhang et al., 2019a), and LookLinear (Balduzzi et al., 2017a). Specifically, we first conduct experiments with 18-layer Transformers on the WMT'14 De-En dataset. In our experiments, we observe that all of ReZero (which does not contain layer normalization), FixUp (which also does not contain layer normalization), and LookLinear (which is incorporated with Post-LN) leads to divergent training. With further analysis, we find that the half-precision training and dropout could destabilize FixUp and ReZero, due to the lack of layer normalization. Simultaneously, we find that even for shadow networks, having an over small reliance on residual branches hurts the model performance, which also supports our intuition. For example, as elaborated in Appendix E, applying ReZero to Transformer-small leads to a 1-2 BLEU score drop on the IWSLT'14 De-En dataset.

6 Related Work

Transformer. Transformer (Vaswani et al., 2017) has led to a series of breakthroughs in various domains (Devlin et al., 2019; Velickovic et al., 2018; Huang et al., 2019; Parmar et al., 2018; Ramachandran et al., 2019). Liu et al. (2020a) show that compared to other architectures, removing the warmup phase is more damaging for Transformers, especially Post-LN. Similarly, it has been found that the original Transformer (referred to as Post-LN) is less robust than its Pre-LN variant (Baevski and Auli, 2019; Nguyen and Salazar, 2019; Wang et al., 2019). Our studies go beyond the existing literature on gradient vanishing (Xiong et al., 2019) and identify an essential factor influencing Transformer training greatly.

Deep Network Initialization. It has been observed that deeper networks can lead to better performance. For example, Dong et al. (2020) find that the network depth players a similar role with the sample number in numerical ODE solvers, which hinders the system from getting more precise results. Many attempts have been made to clear obstacles for training deep networks, including various initialization methods. Based on the independence among initialized parameters, one method is derived and found to be useful to handle the gradient vanishing (Glorot and Bengio, 2010). Similar methods are further developed for ReLU networks (He et al., 2015). He et al. (2016) find that deep network training is still hard even after addressing the gradient vanishing issue and propose residual networks. Balduzzi et al. (2017b) identifies the shattered gradient issue and proposes LookLinear initialization.

On the other hand, although it is observed that scaling residual outputs to smaller values helps to stabilize training (Hanin and Rolnick, 2018; Mishkin and Matas, 2015; Zhang et al., 2019a; Bachlechner et al., 2020; Goyal et al., 2017), there is no systematic analysis on what complicates Transformer training or its underlying connection to the dependency on residual branches. Here, we identify that unbalanced gradients are not the direct cause of the Post-LN instability, recognize the amplification effect, and propose a novel adaptive initialization method.

7 Conclusion

In this paper, we study the difficulties of training Transformers in theoretical and empirical manners. Our study in Section 3 suggests that the gradient vanishing problem is not the root cause of unstable Transformer training. Also, the unbalanced gradient distribution issue is mostly addressed by adaptive optimizers. In Section 4, we reveal the root cause of the instability to be the strong dependency on residual branches, which amplifies the fluctuation caused by parameter changes and destabilizes model training. In light of our analysis, we propose Admin, an adaptive initialization method to stabilize Transformers training. It controls the dependency at the beginning of training and maintains the flexibility to capture those dependencies once training stabilizes. Extensive experiments verify our intuitions and show that, without introducing additional hyper-parameters, Admin achieves more stable training, faster convergence, and better performance.

Our work opens up new possibilities to not only further push the state-of-the-art but understand deep network training better. It leads to many interesting future works, including generalizing Theorem 2 to other models, designing new algorithms to automatically adapt deep networks to different training configurations, upgrading the Transformer architecture, and applying our proposed Admin to conduct training in a larger scale.

Acknowledge

We thank all reviewers for their constructive comments; Chengyu Dong, Haoming Jiang, Jingbo Shang, Xiaotao Gu, and Zihan Wang for valuable discussions and comments; Jingbo Shang for sharing GPU machines; and Microsoft for setting up GPU machines. The research was sponsored in part by DARPA No. W911NF-17-C-0099 and No. FA8750-19-2-1004, National Science Foundation IIS-19-56151, IIS-17-41317, IIS 17-04532, and IIS 16-18481, and DTRA HDTRA11810026.

References

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Thomas C. Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian J. McAuley. 2020. Rezero is all you need: Fast convergence at large depth. *ArXiv*, abs/2003.04887.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *ICLR*.

David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. 2017a. The shattered gradients problem: If resnets are the answer, then what is the question? In *ICML*.

David Balduzzi, Marcus Frean, Lennox Leary, J P Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. 2017b. The shattered gradients problem: If resnets are the answer, then what is the question? In *ICML*.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Workshop on Statistical Machine Translation*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Michael Schuster, Zhi-Feng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Chengyu Dong, Liyuan Liu, Zichao Li, and Jingbo Shang. 2020. Towards adaptive residual network training: A neural-ode perspective. In *ICML*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677.

Boris Hanin and David Rolnick. 2018. How to start training: The effect of initialization and architecture. In *NeurIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer: Generating music with long-term structure. In *ICLR*.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *NeurIPS*.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020a. On the variance of the adaptive learning rate and beyond. In *ICLR*.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020b. Very deep transformers for neural machine translation. *ArXiv*, abs/2008.07772.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2020. Understanding and improving transformer from a multi-particle dynamic system point of view. In *ICLR Workshop DeepDiffEq*.

Dmytro Mishkin and Juan E. Sala Matas. 2015. All you need is a good init. In *ICLR*.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *IWSLT*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT Demonstrations*.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *ICML*.

Martin Popel and Ondrej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110:43 – 70.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. 2019. Stand-alone self-attention in vision models. In *NeurIPS*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ArXiv*, abs/1312.6120.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

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

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019a. Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Lijun Wu, Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019b. Depth growing for neural machine translation. In *ACL*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shu xin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Li-Wei Wang, and Tie-Yan Liu. 2019. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. 2019a. Fixup initialization: Residual learning without normalization. In *ICLR*.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J. Reddi, Surinder Kumar, and Suvrit Sra. 2019b. Why adam beats sgd for attention models. *ArXiv*, abs/1912.03194.

Guangxiang Zhao, Xu Sun, Jingjing Xu, Zhiyuan Zhang, and Liangchen Luo. 2019. Muse: Parallel multi-scale attention for sequence to sequence learning. *ArXiv*, abs/1911.09483.

Appendices

A Gradients at Initialization

Here, we first reveal that Pre-LN does not suffer from the gradient vanishing. Then we establish that only the Post-LN decoder suffers from the gradient vanishing, but not the Post-LN encoder. For simplicity, we use $\Delta\mathbf{x}$ to denote gradients, *i.e.*, $\Delta\mathbf{x} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ where $\mathcal{L}$ is the training objective. Following the previous study (Bengio et al., 1994; Glorot and Bengio, 2010; He et al., 2015; Saxe et al., 2013), we analyze the gradient distribution at the very beginning of training, assume that the randomly initialized parameters and the partial derivative with regard to module inputs are independent.

A.1 Pre-LN Analysis

For Pre-LN encoders, we have $\mathbf{x}^{(pe)}_{2i} = \mathbf{x}^{(pe)}_{2i-1} + f_{\text{FFN}}(f_{\text{LN}}(\mathbf{x}^{(pe)}_{2i-1}))$ and $\Delta\mathbf{x}^{(pe)}_{2i-1} = \Delta\mathbf{x}^{(pe)}_{2i}(1 + \frac{\partial f_{\text{FFN}}(f_{\text{LN}}(\mathbf{x}^{(pe)}_{2i-1}))}{\partial\mathbf{x}^{(pe)}_{2i}})$. At initialization, the two terms on the right part are approximately independent and $E[\frac{\partial f_{\text{FFN}}(f_{\text{LN}}(\mathbf{z}^{(pe)}_{2i-1}))}{\partial\mathbf{x}^{(pe)}_{2i}}] = 0$. Therefore we have $\text{Var}[\Delta\mathbf{x}^{(pe)}_{2i-1}] \geq \text{Var}[\Delta\mathbf{x}^{(pe)}_{2i}]$. Similarly, we can get $\text{Var}[\Delta\mathbf{x}^{(pe)}_{2i-2}] \geq \text{Var}[\Delta\mathbf{x}^{(pe)}_{2i-1}]$ thus $\forall i \leq j, \text{Var}[\Delta\mathbf{x}^{(pe)}_{i}] \geq \text{Var}[\Delta\mathbf{x}^{(pe)}_{j}]$. Applying the same analysis to Pre-LN decoders, we can get $\forall i \leq j, \text{Var}[\Delta\mathbf{x}^{(pd)}_{i}] \geq \text{Var}[\Delta\mathbf{x}^{(pd)}_{j}]$. Thus, lower layers have larger gradients than higher layers, and gradients do not vanish in the backpropagation.

REMARK 1. — For Pre-LN, if $\forall i, \Delta\mathbf{x}^{(p\cdot)}_{i}$ and the derivatives of modules in the i-th sub-layer are independent, then $\forall i \leq j, \text{Var}[\Delta\mathbf{x}^{(p\cdot)}_{i}] \geq \text{Var}[\Delta\mathbf{x}^{(p\cdot)}_{j}]$.

A.2 Post-LN Encoder Analysis

Different from Pre-LN, $\mathbf{x}^{(oe)}_{i}$ and $\mathbf{x}^{(oe)}_{i-1}$ are associated with not only the residual connection but the layer normalization, which makes it harder to establish the connection on their gradients. After making assumptions on the model initialization, we find that lower layers in Post-LN encoder also have larger gradients than higher layers, and gradients do not vanish in the backpropagation through the encoder.

THEOREM 1. — For Post-LN Encoders, if $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ in the Layer Norm are initialized as 1 and 0 respectively; all other parameters are initialized by symmetric distributions with zero mean; $\mathbf{x}^{(oe)}_{i}$ and $\Delta\mathbf{x}^{(oe)}_{i}$ are subject to symmetric distributions with zero mean; the variance of $\mathbf{x}^{(oe)}_{i}$ is 1 (*i.e.*, normalized by Layer Norm); $\Delta\mathbf{x}^{(oe)}_{i}$ and the derivatives of modules in i-th sub-layer are independent, we have $\text{Var}[\Delta\mathbf{x}_{i-1}] \geq \text{Var}[\Delta\mathbf{x}_{i}]$.

Proof. We first prove $\text{Var}[\Delta\mathbf{x}^{(oe)}_{2i-1}] \geq \text{Var}[\Delta\mathbf{x}^{(oe)}_{2i}]$, *i.e.*, the backpropagation through FFN sublayers does not suffer from gradient vanishing. In Post-LN encoders, the output of FFN sublayers is calculated as $\mathbf{x}^{(oe)}_{2i} = f_{\text{LN}}(\mathbf{b}^{(oe)}_{2i})$ where $\mathbf{b}^{(oe)}_{2i} = \mathbf{x}^{(oe)}_{2i-1} + \max(0, \mathbf{x}^{(oe)}_{2i-1}W^{(1)})W^{(2)}$. Since at initialization, $W^{(1)}$ and $W^{(2)}$ are independently randomized by symmetric distributions, we have $\mathbb{E}[\mathbf{b}^{(oe)}_{2i}] = 0$ and

$$\mathbf{x}^{(oe)}_{2i} = \frac{\mathbf{x}^{(oe)}_{2i-1} + \max(\mathbf{x}^{(oe)}_{2i-1}W^{(1)}, 0)W^{(2)}}{\sigma_{b,2i}}$$

where $\sigma^2_{b,2i} = \text{Var}[\mathbf{b}^{(oe)}_{2i}]$. Referring to the dimension of $W^{(1)}$ as $D \times D_f$, He et al. (2015) establishes that

$$\text{Var}[\max(\mathbf{x}^{(oe)}_{2i-1}W^{(1)}, 0)W^{(2)}] = \frac{1}{2}DD_f \text{Var}[w^{(1)}] \text{Var}[w^{(2)}] \text{Var}[\mathbf{x}^{(oe)}_{2i-1}].$$

Since in Post-LN, $\mathbf{x}^{(oe)}_{2i-1}$ is the output of layer norm, we have $\text{Var}[\mathbf{x}^{(oe)}_{2i-1}] = 1$. Thus,

$$\begin{aligned}\sigma^2_{b,2i} &= \text{Var}[\mathbf{b}^{(oe)}_{2i}] = \text{Var}[\mathbf{x}^{(oe)}_{2i-1}] + \text{Var}[\max(\mathbf{x}^{(oe)}_{2i-1}W^{(1)}, 0)W^{(2)}] \\ &= 1 + \frac{1}{2}DD_f \text{Var}[w^{(1)}] \text{Var}[w^{(2)}]. \quad (1)\end{aligned}$$

Assuming different terms are also independent in the backpropagation, we have

$$\mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}] \geq \mathrm{Var}[\frac{1}{\sigma_{b,2i}}(\Delta \mathbf{x}_{2i}^{(oe)} + \Delta \mathbf{x}_{2i}^{(oe)} \frac{\partial \max(\mathbf{x}_{2i-1}^{(oe)} W^{(1)}, 0) W^{(2)}}{\partial \mathbf{x}_{2i-1}^{(oe)}})].$$

At initialization, He et al. (2015) establishes that

$$\mathrm{Var}[\Delta \mathbf{x}_{2i}^{(oe)} \frac{\partial \max(\mathbf{x}_{2i-1}^{(oe)} W^{(1)}, 0) W^{(2)}}{\partial \mathbf{x}_{2i-1}^{(oe)}}] = \frac{1}{2} D D_f \mathrm{Var}[w^{(1)}] \mathrm{Var}[w^{(2)}] \mathrm{Var}[\Delta \mathbf{x}_{2i}^{(oe)}].$$

Therefore, we have

$$\mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}] \geq \frac{1}{\sigma_{b,2i}^2}(1 + \frac{1}{2} D D_f \mathrm{Var}[w^{(1)}] \mathrm{Var}[w^{(2)}]) \mathrm{Var}[\Delta \mathbf{x}_{2i}^{(oe)}]. \tag{2}$$

Combining Equation 1 with Equation 2, we have

$$\mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}] \geq \mathrm{Var}[\Delta \mathbf{x}_{2i}^{(oe)}] \tag{3}$$

which shows the backpropagation through FFN sublayers does not suffer from gradient vanishing.

Now we proceed to prove that, $\mathrm{Var}[\Delta \mathbf{x}_{2i-2}^{(oe)}] \geq \mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}]$, *i.e.*, the backpropagation through Self-Attention sublayers do not suffer from gradient vanishing. In Post-LN encoders, the output of Self-Attention sublayers are calculated as $\mathbf{x}_{2i-1}^{(oe)} = f_{\mathrm{LN}}(\mathbf{b}_{2i-1}^{(oe)})$ where $\mathbf{b}_{2i-1}^{(oe)} = \mathbf{x}_{2i-2}^{(oe)} + \mathbf{a}_{2i-1}^{(oe)}$ and $\mathbf{a}_{2i-1}^{(od)} = \sum_h f_s(\mathbf{x}_{2i-2}^{(oe)} W_h^{(Q)} W_h^{(K)} \mathbf{x}^{T(oe)}_{2i-2}) \mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}$. At initialization, since $W^{(Q)}$, $W^{(K)}$, $W^{(V_1)}$, and $W^{(V_2)}$ are independently randomized by symmetric distributions, we have $\mathbb{E}[\mathbf{b}_{2i-1}^{(od)}] = 0$, thus $\mathbf{x}_{2i-1}^{(oe)} = \frac{\mathbf{b}_{2i-1}^{(oe)}}{\sigma_{b,2i-1}}$, where $\sigma_{b,2i-1}^2 = \mathrm{Var}[\mathbf{b}_{2i-1}^{(oe)}] = \mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)}] + \mathrm{Var}[\mathbf{a}_{2i-1}^{(oe)}]$.

Referring $\mathbb{E}[{f_s}^2(\mathbf{x}_{2i-2}^{(oe)} W_h^{(Q)} W_h^{(K)} \mathbf{x}^{T(oe)}_{2i-2})]$ as P_h, we have

$$\mathrm{Var}[\mathbf{a}_{2i-1}^{(od)}] = \mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}] H P_h.$$

Similar to He et al. (2015), we have

$$\mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}] = \frac{D^2}{H} \mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)}] \mathrm{Var}[w^{(V_1)}] \mathrm{Var}[w^{(V_2)}].$$

Since $\mathbf{x}_{2i-2}^{(oe)}$ is the output of layer norm, we have $\mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)}] = 1$. Thus,

$$\sigma_{b,2i-1}^2 = 1 + D^2 P_h \mathrm{Var}[\mathbf{x}_{2i-2}^{(oe)}] \mathrm{Var}[w^{(V_1)}] \mathrm{Var}[w^{(V_2)}]. \tag{4}$$

In the backpropagation, we have

$$\begin{aligned}
\mathrm{Var}[\Delta \mathbf{x}_{2i-2}^{(oe)}] &\geq \mathrm{Var}[\frac{1}{\sigma_{b,2i-1}}(\Delta \mathbf{x}_{2i-1}^{(oe)} + \Delta \mathbf{x}_{2i-1}^{(oe)} \sum_h \frac{\partial f_s(\mathbf{x}_{2i-2}^{(oe)} W_h^{(Q)} W_h^{(K)} \mathbf{x}^{T(oe)}_{2i-2}) \mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}}{\partial \mathbf{x}_{2i-2}^{(oe)}})] \\
&\geq \frac{1}{\sigma_{b,2i-1}^2}(\mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}] + \mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)} \sum_h f_s(\mathbf{x}_{2i-2}^{(oe)} W_h^{(Q)} W_h^{(K)} \mathbf{x}^{T(oe)}_{2i-2}) \frac{\partial \mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}}{\partial \mathbf{x}_{2i-2}^{(oe)}}])
\end{aligned}$$

At initialization, we assume $\Delta \mathbf{x}_{2i-1}^{(oe)}$ and model parameters are independent (He et al., 2015), thus

$$\begin{aligned}
&\mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)} \sum_h f_s(\mathbf{x}_{2i-2}^{(oe)} W_h^{(Q)} W_h^{(K)} \mathbf{x}^{T(oe)}_{2i-2}) \frac{\partial \mathbf{x}_{2i-2}^{(oe)} W_h^{(V_1)} W_h^{(V_2)}}{\partial \mathbf{x}_{2i-2}^{(oe)}}] \\
=&D^2 P_h \mathrm{Var}[\Delta \mathbf{x}_{2i-1}^{(oe)}] \mathrm{Var}[w^{(V_1)}] \mathrm{Var}[w^{(V_2)}]
\end{aligned}$$

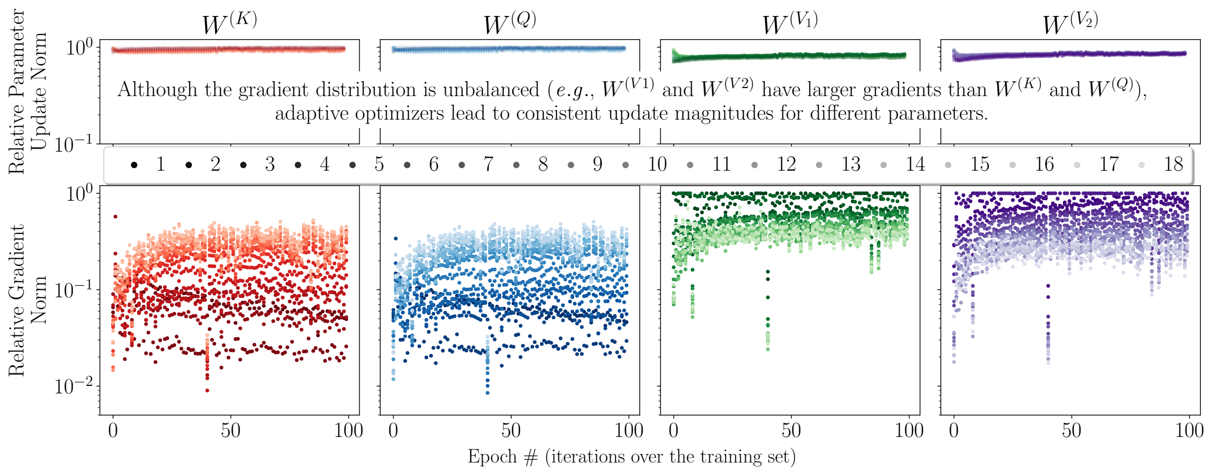


Figure 11: Relative Norm of Gradient (ΔW_i, where W_i is the checkpoint of i-th epoch) and Update ($|W_{i+1} - W_i|$) of Self-Attention Parameters in 12-Layer Pre-LN.

Therefore, we have

$$\mathrm{Var}[\Delta \mathbf{x}^{(oe)}_{2i-2}] \geq \frac{1}{\sigma^2_{b,2i-1}}(1 + D^2 P_h \,\mathrm{Var}[w^{(V_1)}]\,\mathrm{Var}[w^{(V_2)}])\,\mathrm{Var}[\Delta \mathbf{x}^{(oe)}_{2i-1}]. \tag{5}$$

Integrating Equation 4 with Equation 5, we have

$$\mathrm{Var}[\Delta \mathbf{x}^{(oe)}_{2i-2}] \geq \mathrm{Var}[\Delta \mathbf{x}^{(oe)}_{2i-1}]. \tag{6}$$

Combining Equation 3 and Equation 6, we have $\mathrm{Var}[\Delta \mathbf{x}_{i-1}] \geq \mathrm{Var}[\Delta \mathbf{x}_i]$. $\square$

A.3 Post-LN Decoder Analysis

In Post-LN, the Encoder-Attention sub-layer suffers from gradient vanishing. The Encoder-Attention sub-layer calculates outputs as $\mathbf{x}^{(od)}_{3i-1} = f_{\text{LN}}(\mathbf{b}^{(od)}_{3i-1})$ where $\mathbf{b}^{(od)}_{3i-1} = \mathbf{x}^{(od)}_{3i-2} + \mathbf{a}^{(od)}_{3i-1}$ and $\mathbf{a}^{(od)}_{3i-1} = \sum_h f_s(\mathbf{x}^{(od)}_{3i-2} W^{(Q)}_h W^{(K)}_h \mathbf{x}^{T(oe)})\mathbf{x}^{(oe)} W^{(V_1)}_h W^{(V_2)}_h$. Here $\mathbf{x}^{(oe)}$ is encoder outputs and f_s is the row-wise softmax function. In the backpropagation, $\Delta \mathbf{x}^{(od)}_{3i-2} \approx \frac{\Delta \mathbf{x}^{(od)}_{3i-1}}{\sigma_{b,3i-1}}(1 + \frac{\partial \mathbf{a}^{(od)}_{3i-1}}{\mathbf{x}^{(od)}_{3i-2}})$. All of the backpropagations from $\mathbf{a}^{(od)}_{3i-1}$ to $\mathbf{x}^{(od)}_{3i-2}$ went through the softmax function, we have $\mathrm{Var}[\frac{\partial \mathbf{a}^{(od)}_{3i-1}}{\mathbf{x}^{(od)}_{3i-2}}] + 1 \leq \sigma^2_{b,3i-1}$. Thus, those backpropagations suffer from gradient vanishing. This observation is further verified in Figure 3, as the encoder attention bars (gradients of encoder-attention outputs) are always shorter than self-attention bars (gradients of encoder-attention inputs), while adjacent self-attention bars and fully connected bars usually have the same length.

A.4 Distributes of Unbalanced Gradients

As in Figure 5 and Figure 11, the gradient distribution of Attention modules is unbalanced even for Pre-LN. Specifically, parameters within the softmax function (*i.e.*, $W^{(K)}$ and $W^{(V_1)}$) suffer from gradient vanishing (*i.e.*, $\frac{\partial f_s(x_0,\cdots,x_i,\cdots)}{\partial x_i} \leq 1$) and have smaller gradients than other parameters.

With further analysis, we find it is hard to neutralize the gradient vanishing of softmax. Unlike conventional non-linear functions like ReLU or sigmoid, softmax has a dynamic input length (*i.e.*, for the sentences with different lengths, inputs of softmax have different dimensions). Although this setting allows Attention modules to handle sequential inputs, it restricts them from having stable and consistent backpropagation. Specifically, let us consider the comparison between softmax and sigmoid. For the sigmoid function, although its derivation is smaller than 1, this damping effect is consistent for all inputs. Thus, sigmoid can be neutralized by a larger initialization (Glorot and Bengio, 2010). For softmax, its damping effect is different for different inputs and cannot be neutralized by a static initialization.

Also, we observe that adaptive optimizers largely address this issue. Specifically, we calculate the norm of parameter change in consequent epochs (*e.g.*, $|W_{t+1}^{(K)} - W_t^{(K)}|$ where $W_t^{(K)}$ is the checkpoint saved after t epochs) and visualize the relative norm (scaled by the largest value in the same network) in Figure 11. Comparing the relative norm of parameter gradients and parameter updates, we notice that: although the gradient distribution is unbalanced, adaptive optimizers successfully assign different learning rates to different parameters and lead to consistent update magnitudes. This result explains why the vanilla SGD fails for training Transformer (*i.e.*, lacking the ability to handle unbalanced gradient distributions). Besides, it implies that the unbalanced gradient distribution (*e.g.*, gradient vanishing) has been mostly addressed by adaptive optimizers and may not significantly impact the training instability.

B Proof of Theorem 2

Here, we elaborate the derivation for Theorem 2, which establishes the relationship between layer number and output fluctuation brought by parameter change.

THEOREM 2. — Consider a N-layer Transformer $\widehat{\mathbf{x}} = \mathcal{F}(\widehat{\mathbf{x}}_0, W)$, where $\widehat{\mathbf{x}}_0$ is the input and W is the parameter. If the layer dependency stays the same after a parameter change (*i.e.*, $\beta_{i,j}$ has the same value after changing W to W^*, where W is randomly initialized and $\delta = W^* - W$ is independent to W), the output change (*i.e.*, $\mathrm{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)]$) can be estimated as $\sum_{i=1}^N \beta_{i,i}^2 C$ where C is a constant.

Proof. We refer the module in i sub-layer as $\mathbf{a}_i = \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i)$, where $\widehat{\mathbf{x}}_i = \sum_{j\le i} \beta_{i,j}\widehat{\mathbf{a}}_j$ is the normalized residual output and $\widehat{\mathbf{a}}_i = \frac{\mathbf{a}_i}{\sqrt{\mathrm{Var}\,\mathbf{a}_i}}$ is the normalized module output. The final output is marked as $\widehat{\mathbf{x}} = \mathcal{F}(\mathbf{x}_0, W) = \sum_{j\le N} \beta_{N,j}\widehat{\mathbf{a}}_j$. To simplify the notation, we use the superscript $*$ to indicate variables related to W^*, *e.g.*, $\widehat{\mathbf{x}}^* = \mathcal{F}(\mathbf{x}_0, W^*)$ and $\mathbf{a}_i^* = \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)$.

At initialization, all parameters are initialized independently. Thus $\forall i \ne j$, $\widehat{\mathbf{a}}_i$ and $\widehat{\mathbf{a}}_j$ are independent and $1 = \mathrm{Var}[\sum_{j\le i} \beta_{i,j}\widehat{\mathbf{a}}_j] = \sum_{j\le i}\beta_{i,j}^2$. Also, since k-layer and $(k+1)$-layer share the residual connection to previous layers, $\forall i, j \le k$ we have $\frac{\beta_{i,k}}{\beta_{j,k}} = \frac{\beta_{i,k+1}}{\beta_{j,k+1}}$. Thus $\forall i \le k$, $\beta_{i,k+1}^2 = (1-\beta_{k,k}^2)\beta_{i,k}^2$ and

$$\begin{aligned}\mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*] &= \mathrm{Var}[\sum_{j\le i}\beta_{i,j}(\widehat{\mathbf{a}}_j - \widehat{\mathbf{a}}_j^*)] = \sum_{j\le i}\beta_{i,j}^2\,\mathrm{Var}[\widehat{\mathbf{a}}_j - \widehat{\mathbf{a}}_j^*]\\ &= \beta_{i,i}^2\,\mathrm{Var}[\widehat{\mathbf{a}}_i - \widehat{\mathbf{a}}_i^*] + (1-\beta_{i,i}^2)\,\mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*]. \qquad (7)\end{aligned}$$

Now, we proceed to analyze $\mathrm{Var}[\widehat{\mathbf{a}}_i - \widehat{\mathbf{a}}_i^*]$. Specifically, we have

$$\begin{aligned}\mathrm{Var}[\widehat{\mathbf{a}}_i - \widehat{\mathbf{a}}_i^*] &= \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)]\\ &= \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i) + \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)]\\ &= \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i)] + \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)]. \qquad (8)\end{aligned}$$

Since W is randomly initialized, $\mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)]$ should have the same value for all layers, thus we use a constant C to refer its value ($C = \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i^*)]$ and $C \approx |\delta|\cdot|\nabla\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i)|$). As to $\mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i)]$, since the sub-layer of Transformers are mostly using linear weights with ReLU nonlinearity and $1 = \mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i)] = \mathrm{Var}[\widehat{\mathbf{x}}_{i-1}]$, we have $\mathrm{Var}[\mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}, W_i) - \mathcal{G}_i(\widehat{\mathbf{x}}_{i-1}^*, W_i)] \approx \mathrm{Var}[\widehat{\mathbf{x}}_{i-1} - \widehat{\mathbf{x}}_{i-1}^*]$. Thus, we can rewrite Equation 8 and get

$$\mathrm{Var}[\widehat{\mathbf{a}}_i - \widehat{\mathbf{a}}_i^*] \approx \mathrm{Var}[\widehat{\mathbf{x}}_{i-1} - \widehat{\mathbf{x}}_{i-1}^*] + C$$

With Equation 7, we have

$$\begin{aligned}\mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*] &= \beta_{i,i}^2\,\mathrm{Var}[\widehat{\mathbf{a}}_i - \widehat{\mathbf{a}}_i^*] + (1-\beta_{i,i}^2)\,\mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*]\\ &\approx \beta_{i,i}^2(\mathrm{Var}[\widehat{\mathbf{x}}_{i-1} - \widehat{\mathbf{x}}_{i-1}^*] + C) + (1-\beta_{i,i}^2)\,\mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*]\\ &= \mathrm{Var}[\widehat{\mathbf{x}}_i - \widehat{\mathbf{x}}_i^*] + \beta_{i,i}^2 C\end{aligned}$$

Therefore, we have $\mathrm{Var}[\mathcal{F}(\mathbf{x}_0, W) - \mathcal{F}(\mathbf{x}_0, W^*)] \approx \sum_{i=1}^N \beta_{i,i}^2 C$. □

C Admin Implementation Details

As introduced in Section 4.3, we introduce a new set of parameters to rescale the module outputs. Specifically, we refer these new parameters as ω and construct the Post-LN sub-layer as:

$$\mathbf{x}_i = f_{\text{LN}}(\mathbf{b}_i), \text{where } \mathbf{b}_i = \mathbf{x}_{i-1} \cdot \boldsymbol{\omega}_i + f_i(\mathbf{x}_{i-1})$$

where $\cdot$ is the element-wise product.

After training, Admin can be reparameterized as the conventional Post-LN structure (*i.e.*, removing $\boldsymbol{\omega}_i$). Specifically, we consider $\mathbf{x}_i = \frac{\mathbf{b}_i}{\sigma_b}\gamma + \boldsymbol{\nu}$. Then, for feedforward sub-layers, we have

$$\mathbf{b}_i = \mathbf{x}_{i-1} \cdot \omega + \max(0, \mathbf{x}_{i-1}W^{(1)})W^{(2)}, \text{where } \mathbf{x}_i = \frac{\mathbf{b}_{i-1}}{\sigma_b}\gamma + \boldsymbol{\nu}.$$

It can be reparameterized by changing γ, $\boldsymbol{\nu}$, $W^{(1)}$ to $\gamma\boldsymbol{\omega}_i$, $\boldsymbol{\nu}\boldsymbol{\omega}_i$, $\frac{1}{\boldsymbol{\omega}_i}W^{(1)}$ respectively, *i.e.*,

$$\mathbf{b}'_i = \mathbf{x}'_{i-1} + \max(0, \mathbf{x}'_{i-1}\frac{1}{\boldsymbol{\omega}_i}W^{(1)})W^{(2)}, \text{where } \mathbf{x}'_{i-1} = \frac{\mathbf{b}'_{i-1}}{\sigma_b}\gamma\boldsymbol{\omega}_i + \boldsymbol{\nu}\boldsymbol{\omega}_i.$$

For Self-Attention sub-layers, we have

$$\mathbf{b}_i = \mathbf{x}_{i-1} + \sum_h f_s(\mathbf{x}_{i-1}W_h^{(Q)}W_h^{(K)}\mathbf{x}_{i-1})\mathbf{x}_{i-1}W_h^{(V_1)}W_h^{(V_2)}, \text{where } \mathbf{x}_i = \frac{\mathbf{b}_{i-1}}{\sigma_b}\gamma + \boldsymbol{\nu}.$$

It can be reparameterized by changing γ, $\boldsymbol{\nu}$, $W_h^{(Q)}$, $W_h^{(K)}$, $W_h^{(V_1)}$ to $\gamma\boldsymbol{\omega}_i$, $\boldsymbol{\nu}\boldsymbol{\omega}_i$, $\frac{1}{\boldsymbol{\omega}_i}W_h^{(Q)}$, $\frac{1}{\boldsymbol{\omega}_i}W_h^{(K)}$ $\frac{1}{\boldsymbol{\omega}_i}W_h^{(V_1)}$ respectively, *i.e.*,

$$\mathbf{b}'_i = \mathbf{x}'_{i-1} + \sum_h f_s(\mathbf{x}'_{i-1}\frac{1}{\boldsymbol{\omega}_i}W_h^{(Q)}W_h^{(K)}\frac{1}{\boldsymbol{\omega}_i}\mathbf{x}'_{i-1})\mathbf{x}'_{i-1}\frac{1}{\boldsymbol{\omega}_i}W_h^{(V_1)}W_h^{(V_2)}, \text{where } \mathbf{x}'_{i-1} = \frac{\mathbf{b}'_{i-1}}{\sigma_b}\gamma\boldsymbol{\omega}_i + \boldsymbol{\nu}\boldsymbol{\omega}_i.$$

For Encoder-Attention sub-layers, we have

$$\mathbf{b}_i = \mathbf{x}_{i-1} + \sum_h f_s(\mathbf{x}_{i-1}W_h^{(Q)}W_h^{(K)}\mathbf{x}^{\cdot e})\mathbf{x}^{\cdot e}W_h^{(V_1)}W_h^{(V_2)}, \text{where } \mathbf{x}_i = \frac{\mathbf{b}_{i-1}}{\sigma_b}\gamma + \boldsymbol{\nu}.$$

It can be reparameterized by changing γ, $\boldsymbol{\nu}$, $W_h^{(Q)}$ to $\gamma\boldsymbol{\omega}_i$, $\boldsymbol{\nu}\boldsymbol{\omega}_i$, $\frac{1}{\boldsymbol{\omega}_i}W_h^{(Q)}$ respectively, *i.e.*,

$$\mathbf{b}'_i = \mathbf{x}'_{i-1} + \sum_h f_s(\mathbf{x}'_{i-1}\frac{1}{\boldsymbol{\omega}_i}W_h^{(Q)}W_h^{(K)}\mathbf{x}^{\cdot e})\mathbf{x}^{\cdot e}\frac{1}{\boldsymbol{\omega}_i}W_h^{(V_1)}W_h^{(V_2)}, \text{where } \mathbf{x}'_{i-1} = \frac{\mathbf{b}'_{i-1}}{\sigma_b}\gamma\boldsymbol{\omega}_i + \boldsymbol{\nu}\boldsymbol{\omega}_i.$$

It is easy to find $\mathbf{b}'_i = \mathbf{b}_i$ in all three situations.

From the previous analysis, it is easy to find that introducing the additional parameter $\boldsymbol{\omega}_i$ is equivalent to rescale some model parameters. In our experiments on IWSLT14 De-En, we find that directly rescaling initialization parameters can get roughly the same performance with introducing $\boldsymbol{\omega}_i$. However, it is not very stable when conducting training in a half-precision manner. Accordingly, we choose to add new parameters $\boldsymbol{\omega}_i$ instead of rescaling parameters.

D Experimental Setup

Our experiments are based on the implementation from the fairseq package (Ott et al., 2019). As to pre-processing, we follow the public released script from previous work (Ott et al., 2019; Lu et al., 2020). For WMT'14 datasets, evaluations are conducted on the provided 'newstest14' file, and more details about them can be found in Bojar et al. (2014). For the IWSLT'14 De-En dataset, more analysis and details can be found in Cettolo et al. (2014).

Table 3: ReZero Performance on IWSLT'14 De-En. Models are Transformer-small w. 6-layer encoder & decoder.

Models	Admin	Post-LN	Pre-LN	ReZero	ReZero+Post-LN
BLEU	35.67±0.15	35.64±0.23	35.50±0.04	33.67±0.14	34.67±0.08

Table 4: Performance and model size on WMT'14 En-Fr (AL-BL refers A-layer encoder & B-layer decoder).

Methods	Param. #	dim($W^{(1)}$) in FFN	Enc#-Dec#	BLEU
T5-Base (Raffel et al., 2019)	220 M	512×2048	6L-6L	41.2
T5-Large (Raffel et al., 2019)	770 M	1024×4096	12L-12L	41.5
T5-3B (Raffel et al., 2019)	3 B	1024×16384	24L-24L	42.6
T5-11B (Raffel et al., 2019)	11 B	1024×65536	24L-24L	43.4
Trans.Big-RNMT+ (Chen et al., 2018)	377 M	1024×8192	6L-6L	41.12
DynamicConv (Wu et al., 2019a)	213 M	1024×4096	7L-7L	43.2
DG-Transformer (Wu et al., 2019b)	264 M	1024×4096	8L-8L	43.27
Prime (Zhao et al., 2019)	252 M	1024×4096	6L-6L	43.48
Pre-LN (60L–12L)	262 M	512×2048	60L-12L	43.10
Admin (60L–12L)	262 M	512×2048	60L-12L	43.80

As to model specifics, we directly adopt Transformer-small configurations on the IWSLT'14 De-En dataset and stacks more layers over the Transformer-base model on the WMT'14 En-De and WMT'14 En-Fr datasets. Specifically, on the IWSLT'14 De-En dataset, we use word embedding with 512 dimensions and 6-layer encoder/decoder with 4 heads and 1024 feedforward dimensions; on the WMT'14 En-De and WMT'14 En-Fr datasets, we use word embedding with 512 dimension and 8-head encoder/decoder with 2048 hidden dimensions. Label smoothed cross entropy is used as the objective function with an uncertainty $= 0.1$ (Szegedy et al., 2016).

For Model training, we use RAdam as the optimizer (Liu et al., 2020a) and adopt almost all hyper-parameter settings from Lu et al. (2020). Specifically, for the WMT'14 En-De and WMT'14 En-Fr dataset, all dropout ratios (including (activation dropout and attention dropout) are set to 0.1. For the IWSLT'14 De-En dataset, after-layer dropout is set to 0.3, and a weight decay of 0.0001 is used. As to optimizer, we set $(\beta_1, \beta_2) = (0.9, 0.98)$, use inverse sqrt learning rate scheduler with a warmup phrase (8000 steps on the WMT'14 En-De/Fr dataset, and 6000 steps on the IWSLT'14 De-En dataset). The maximum learning rate is set to $1e^{-3}$ on the WMT'14 En-De dataset and $7e^{-4}$ on the IWSLT'14 De-En and WMT'14 En-Fr datasets. We conduct training for 100 epochs on the WMT'14 En-De dataset, 90 epochs on the IWSLT'14 De-En dataset and 50 epochs on the WMT'14 En-Fr dataset, while the last 10 checkpoints are averaged before inference.

On the IWSLT'14 De-En dataset, we conduct training on one NVIDIA GeForce GTX 1080 Ti GPU and set the maximum batch size to be 4000. On the WMT'14 En-De dataset, we conduct training on four NVIDIA Quadro R8000 GPUs and set maximum batch size (per GPU) as 8196. On the WMT'14 En-Fr dataset, we conduct training with the Nvidia DGX-2 server (6L-6L uses 4 NVIDIA TESLA V100 GPUs and 60L-16L uses 16 NVIDIA TESLA V100 GPUs) and set the maximum batch size (per GPU) as 5000. On the IWSLT'14 De-En dataset, Transformer-small models (w. 37 M Param.) take a few hours to train. On the WMT'14 En-De dataset, 6L-6L models (w. 63 M Param.) take $\sim$ 1 day to train, 12L-12L (w. 107M Param.) models take $\sim$ 2 days to train, and 18L-18L (w. 151M Param.) models take $\sim$ 3 days to train. On the WMT'14 En-Fr dataset, 6L-6L models (w. 67 M Param.) takes $\sim$ 2 days to train, and 60L-12L models (w. 262M Param.) takes $\sim$ 2.5 days to train. All training is conducted in half-precision with dynamic scaling (with a 256-update scaling window and a 0.03125 minimal scale). All our implementations and pre-trained models would be released publicly.

E Comparison to ReZero

Here, we first conduct comparisons with ReZero (Bachlechner et al., 2020) under two configurations– the first employs the original ReZero model, and the second adds layer normalizations in a Post-LN manner. As summarized in Table 3, the ReZero initialization leads to a performance drop, no matter layer normalization is used or not. It verifies our intuition that over small dependency restricts the model potential. At the same time, we find that adding layer normalization to ReZero helps to improve the performance. Intuitively, as dropout plays a vital role in regularizing Transformers, layer normalization helps to not only stabilize training but alleviate the impact of turning off dropouts during the inference.

F Performance on the WMT'14 En-Fr

To explore the potential of Admin, we conduct experiments with 72-layer Transformers on the WMT'14 En-Fr dataset (with a 60-layer encoder and 12-layer decoder, we add less layers to decoder to encourage the model to rely more on the source context).

As in Table 4, Admin (60L–12L) achieves a BLEU score of 43.80, the new state-of-the-art on this long-standing benchmark. This model has a 60-layer encoder and a 12-layer decoder, which is significantly deeper than other baselines. Still, since the number of parameters increases in a quadratic speed with regard to hidden dimensions and a linear speed with regard to layer numbers, our model has roughly the same number of parameters with other baselines. It is worth mentioning that Admin even achieves better performance than all variants of pre-trained T5 models, which demonstrates the great potential of our proposed method. Also, Admin achieves a better performance than Pre-LN (60L–12L), which further verifies that the Pre-LN architecture restricts deep models' potential.