Improving Transformer Optimization Through Better Initialization

Xiao Shi Huang [*1,2] **Felipe Pérez** [*1] **Jimmy Ba** [3,2] **Maksims Volkovs** [1]

Abstract

The Transformer architecture has achieved considerable success recently; the key component of the Transformer is the attention layer that enables the model to focus on important regions within an input sequence. Gradient optimization with attention layers can be notoriously difficult requiring tricks such as learning rate warmup to prevent divergence. As Transformer models are becoming larger and more expensive to train, recent research has focused on understanding and improving optimization in these architectures. In this work our contributions are two-fold: we first investigate and empirically validate the source of optimization problems in the encoder-decoder Transformer architecture; we then propose a new weight initialization scheme with theoretical justification, that enables training without warmup or layer normalization. Empirical results on public machine translation benchmarks show that our approach achieves leading accuracy, allowing to train deep Transformer models with 200 layers in both encoder and decoder (over 1000 attention/MLP blocks) without difficulty. Code for this work is available here: https://github.com/layer6ai-labs/T-Fixup.

1. Introduction

The Transformer model proposed by Vaswani et al. (2017), has proven to be a versatile and effective architecture with broad applications in various fields of deep learning, including neural machine translation (Vaswani et al., 2017; Edunov et al., 2018), language modeling (Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020; Beltagy et al., 2020), video captioning and summarization (Bilkhu et al., 2019; Fan et al., 2018; Chen et al., 2018a) and recommender systems (Chen et al., 2019; Sun et al., 2019). Despite the broad applications, optimization in the Transformer models can be notoriously difficult (Popel & Bojar, 2018). Most successful implementations require learning rate warmup, layer normalization, residual connections and large batch size for learning to work. Removing any of these components hampers optimization, and the model often fails to learn (Popel & Bojar, 2018). Learning rate warmup is particularly puzzling. Unlike most deep learning architectures, where learning rate is initialized to a reasonably high value and then annealed as training progresses, Transformers instead require gradual learning rate warmup at the beginning of training. Starting with a high learning rate without warmup breaks optimization, while training with a small learning rate is prohibitively slow.

As many leading Transformer architectures are large and require many GPU hours to train (Yang et al., 2019; Lan et al., 2020), recent research has focused on understanding and improving optimization in these models (Ahmed et al., 2017; Chen et al., 2018b; Nguyen & Salazar, 2019; Aji & Heafield, 2019; Liu et al., 2020). Chen et al. (2018b) found that layer normalization could be the source of the problem, and moving it off the main branch improves optimization. This finding was further supported by follow up work (Nguyen & Salazar, 2019; Wang et al., 2019). In parallel, Zhang et al. (2019b) showed that layer normalization can be fully removed in related ResNet models if appropriate weight initialization is used. Lastly, Liu et al. (2020) focused on the Adam optimizer, demonstrating that during early stages of optimization the inverse of the second moment is proportional to a divergent integral and can lead to unstable updates. Despite significant progress, there have been conflicting conclusions on learning rate warmup from previous work (Ma & Yarats, 2019; Liu et al., 2020).

In this work, we aim to answer the question of why Transformer training can break down without learning rate warmup, and propose a solution. We first show that instability in the Adam optimizer, in part caused by gradient vanishing from layer normalization, causes optimization to break when warmup is not used. We then build on the work of Zhang et al. (2019b), and propose a weight initialization scheme for the Transformer that fully eliminates the need for layer normalization and warmup. By applying this weight initialization, we achieve highly competitive perfor-

[*]Equal contribution [1]Layer 6 AI, Toronto, ON, Canada [2]University of Toronto, Toronto, ON, Canada [3]Vector Institute, Toronto, ON, Canada. Correspondence to: Xiao Shi Huang <gary@layer6.ai>, Felipe Pérez <felipe@layer6.ai>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

mance on neural machine translation tasks, and show that models with 200 layers in *both* encoder and decoder can be trained without difficulty. To the best of our knowledge this is the first study where Transformer models of such depth are successfully trained. In summary our contributions are as follows:

- Investigate and empirically validate the main cause of failure in Transformer optimization when learning rate warmup is not used.
- Derive weight initialization for the Transformer architecture that eliminates the need for layer normalization and warmup.
- Demonstrate that with our weight initialization scheme, very deep Transformer models can be trained without difficulty.
- Conduct extensive empirical evaluation on public neural machine translation benchmarks showing superior performance.

2. Related Work

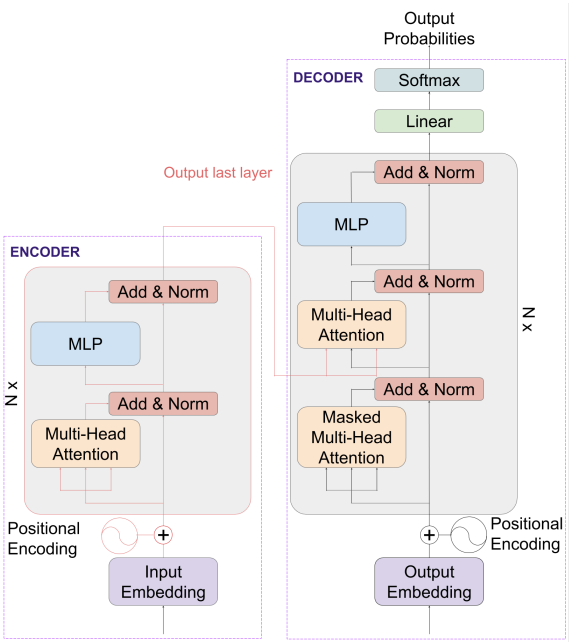


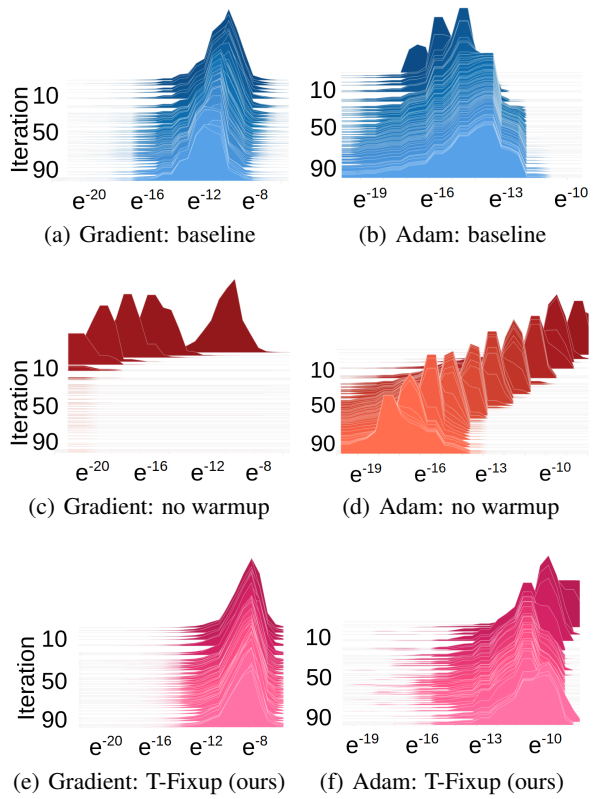
Figure 1. Transformer encoder-decoder architecture with N layers.

We begin by describing the Transformer encoder-decoder architecture originally proposed for neural machine translation (Vaswani et al., 2017). The encoder can be viewed as passing a sequence of input tokens through a series of layers. Each layer is composed of self-attention block followed by MLP block with ReLU activations, and the output of the last layer is typically referred to as "memory". The decoder consists of a series of layers composed of self-attention, encoder-attention using memory as key-value pairs, and MLP blocks. All blocks in encoder and decoder have residual by-pass connections followed by layer normalization. The full Transformer architecture with N layers is shown in Figure 1.

Given the considerable success of Transformer models in multiple areas of deep learning (Vaswani et al., 2017; Yang et al., 2019; Lan et al., 2020), significant effort has recently been devoted to understating, improving and simplifying optimization in these models. Zhang et al. (2019b) showed that architectures with MLP blocks and residual connections can be trained without layer normalization. This is achieved by designing a weight initialization scheme that reduces variance of the error signal during gradient updates. The authors also apply this approach to the Transformer architecture with similar effect. However, to preserve performance multiple tricks and extra layers are added into the model. We expand on the work of Zhang et al. (2019b) and demonstrate that, by incorporating the specifics of the Transformer architecture, we can derive a weight initialization scheme that is highly effective on its own without any additions to the model or optimization procedure.

Another line of research has recently found that the need for the warmup phase arises from the undesirably large variance of the inverse of the second moment in the Adam optimizer during early stages of learning (Liu et al., 2020). Without enough update history, noisy second moment estimates can skew weight updates away from regions where better minimums exist. The authors correct this by designing a rectified version of the Adam optimizer and demonstrate better performance. However, this analysis should apply to any model trained with Adam; yet only the Transformer and its derivative models need the warmup phase, while most other models can be successfully trained without it. Moreover, shallow Transformers with three layers or less can be successfully trained without warmup (Nguyen & Salazar, 2019). Collectively, these findings indicate that while Adam could be part of the problem, the Transformer architecture amplifies it and can even be the root cause.

In parallel, Chen et al. (2018b) observed that the warmup requirement can be lifted by using the Pre-LN variant of the Transformer, where layer normalization is moved inside each block and applied before other operations. Similar findings were reported by Nguyen & Salazar (2019), who additionally experiment with different types of normalizations such as ScaleNorm and FixNorm to speed up convergence and improve performance. Wang et al. (2019) showed that Pre-LN and additional residual connections also help with the optimization of deep Transformer models that have up to 30 layers. Finally, Zhang et al. (2019a) utilized a depth-dependent scaling initialization together with merged attention in the decoder to improve training of Transformer models with up to 20 layers.



(a) Gradient: baseline (b) Adam: baseline

(c) Gradient: no warmup (d) Adam: no warmup

(e) Gradient: T-Fixup (ours) (f) Adam: T-Fixup (ours)

Figure 2. Log-scale histograms of gradients (left) and Adam updates (right) for the input embedding layer in the encoder during the first 100 steps of training. Baseline uses warmup and layer normalization, removing warmup makes gradients vanish destabilizing Adam. Our T-Fixup initialization eliminates the need for warmup and layer normalization, while keeping both gradients and Adam updates stable throughout learning.

In this paper we combine the findings from previous work, and show that the layer normalization does indeed cause problems in Transformer optimization. However, the gradient doesn't explode as hypothesised by Xiong et al. (2020) but rather vanishes for lower layer parameters, particularly for input embeddings. This is caused by a combination of the large variance of the Adam updates as noted by Liu et al. (2020), and gradient propagation through layer normalization. We then build on the work of Zhang et al. (2019b) and derive a weight initialization scheme that eliminates the need for layer normalization all together, and as a result makes optimization work without learning rate warmup.

3. Our Approach

We first identify and empirically validate the source of the problem in Transformer optimization that makes learning rate warmup necessary. We then derive a weight initialization scheme that eliminates this problem, and demonstrate that with this initialization deep Transformer models can be trained without difficulty.

3.1. Problem in Transformer Optimization

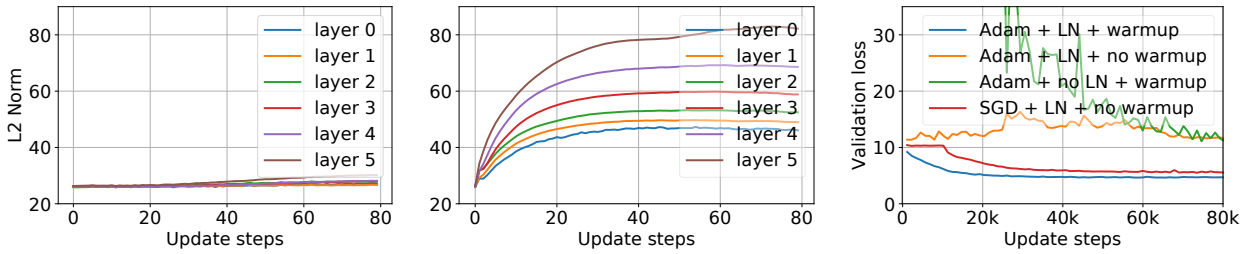
In this section we demonstrate that the requirement for warmup comes from a combined effect of high variance in the Adam optimizer and backpropagation through layer normalization. Liu et al. (2020) showed that at the beginning of training the variance of the inverse second moment is proportional to a divergent integral. This can lead to problematic updates early on and significantly affect optimization. The authors further suggested that this is the source of problems when training Transformers, but didn't identify how exactly it occurs. In parallel, Xiong et al. (2020) found that the magnitude of error signal backpropagating through layer normalization is inversely proportional to magnitude of the input. Specifically, the gradient has the following property:

$$\left\| \frac{\partial \mathrm{LN}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\| = O\left(\frac{\sqrt{d}}{||\boldsymbol{x}||} \right) \tag{1}$$

where $\boldsymbol{x}$ is the input to layer normalization and d is the embedding dimension. If input norm $||\boldsymbol{x}||$ is larger than $\sqrt{d}$ then backpropagation through layer normalization has a down scaling effect that reduces gradient magnitude for lower layers. Compounding across multiple layers this can quickly lead to gradient vanishing.

Combining these two findings, we find that when Transformer is trained without learning rate warmup, the variance in the Adam optimizer, amplified by large initial learning rate, leads to large updates at the beginning of training. Figure 2 shows log-scale histograms for input embedding gradients and Adam updates during the first 100 steps of training for models trained with and without warmup. Comparing Figures 2(b) and 2(d) we see that Adam updates are an order of magnitude larger during early stages of training for model without warmup. Large updates in turn increase the magnitude of inputs to layer normalization as seen in Figure 3. Figures 3(a) and 3(b) show average L2 norm of the input to layer normalization after self-attention block in each layer of the decoder for models trained, with and without warmup respectively. We see that the L2 norm increases 2x to 3x particularly in the upper layers when warmup is not used. The embedding dimension is set to $d = 512$ ($\sqrt{d} \approx 22.6$) so $\frac{\sqrt{d}}{||\boldsymbol{x}||}$ quickly becomes significantly smaller than 1 as training progresses. This effect is especially pronounced in the upper layers so lower layers get progressively smaller gradient. Figure 2(c) demonstrates that after only a few steps the gradient to the input embeddings (lowest layer) vanishes completely.

Jointly, these effects create a cycle where large updates from instability in the Adam optimizer coupled with increased learning rate, lead to gradient vanishing through layer normalization. Vanishing gradients then further destabilize the



(a) Warmup (b) No warmup (c) Validation curves

Figure 3. 3(a) and 3(b) show average L2 norms of the input to layer normalization after self-attention block in each layer of the decoder for models trained with and without warmup respectively. 3(c) shows validation curves for models trained with different settings of Adam/SGD optimizer, layer normalization (LN) and earning rate warmup. Adam + LN + warmup corresponds to the original baseline proposed by Vaswani et al. (2017). All results are for the Transformer small model on the IWSLT'17 De-En dataset.

inverse of the second moment in the Adam update. This cycle continues until there is enough history to stabilize the optimizer, but by that time the model is stuck in a bad plateau (Figure 2(c)). The destabilizing effect of the Adam optimizer can be further observed in Figure 3(c). Here, we show validation curves for models trained with various combinations of Adam/SGD optimizer, layer normalization and warmup. All modes have identical configurations and only differ in these three settings. We see that when Adam is used without warmup, training fully diverges and no progress is made. However, when Adam is replaced with SGD the model gets stuck initially, but then recovers and is able to converge. This further supports the conclusion that the instability in the Adam optimizer, due in part to vanishing gradients, requires gradual learning rate warmup for optimization to work. Replacing Adam with SGD however, makes training significantly slower and is not a feasible solution particularly for large models.

Another potential solution that has been proposed is to move layer normalization off the main branch (Pre-LN). This reduces gradient vanishing in lower layers and stabilizes Adam updates, making optimization easier especially for deeper Transformer models (Chen et al., 2018b; Wang et al., 2019). However, recent work has indicated that Pre-LN can degrade performance on neural machine translation (NMT) tasks (Nguyen & Salazar, 2019; Wang et al., 2019). For completeness we also try to remove layer normalization from the model. But, as seen from Figure 3(c), this makes training very unstable even when warmup is used. In this work we show that with appropriate weight initialization layer normalization can be successfully removed without destabilizing training.

3.2. Fix Through Initialization

To address the challenges in Transformer optimization we develop an initialization scheme that enables warmup-free training even for very deep models. We discussed in Section 3.1 that part of the problem comes from the unstable early updates in the Adam optimizer. The original Adam update uses a moving average $\mathbf{v}_t = \beta\mathbf{v}_{t-1} + (1-\beta)\mathbf{g}_t^2$ to estimate the second moment of the gradients $\mathbf{g}_t$. The learning rate is then scaled by the inverse square root of $\mathbf{v}_t$. Liu et al. (2020) showed that the variance of this update is unbounded. This can be particularly problematic in early stages when second moment is estimated from only a few samples.

Appropriately bounding $\mathbf{v}_t$ would ensure that the variance is finite and reduce instability. We can obtain such a bound by requiring that the update at each step is limited in magnitude. Zhang et al. (2019b) demonstrated that this can be achieved with the SGD optimizer for functions composed of residual MLP/convolution blocks by appropriately initializing them. We build on this work and derive analogous initialization for the attention blocks that are the main components of the Transformer architecture. We show theoretical derivation for the SGD update, and demonstrate empirically that our approach also works well for Adam.

We use $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ to denote the Transformer model with the output softmax layer and all layer normalization blocks removed, and $\mathcal{L}$ denotes the target loss function. Here, $\boldsymbol{x}$ and $\boldsymbol{y}$ are inputs to encoder and decoder respectively, and $\boldsymbol{\theta}$ are free parameters to be learned. Analogous to Zhang et al. (2019b), given a learning rate η, we set the following optimization goal for the SGD update:

GOAL: $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ is updated by $\Theta(\eta)$ per optimization step as $\eta \to 0$. That is, $||\Delta f|| = \Theta(\eta)$, where $\Delta f \triangleq f\left(\boldsymbol{x} - \eta\frac{\partial\mathcal{L}}{\boldsymbol{x}}, \boldsymbol{y} - \eta\frac{\partial\mathcal{L}}{\boldsymbol{y}}; \boldsymbol{\theta} - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}\right) - f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$.

The goal requires each update to be bounded in magnitude independent of model depth. We give an overview of how to obtain such a bound here and refer the reader to the Appendix for detailed derivation. We first separate the Transformer into encoder f_e and decoder f_d. The full

model can be written as $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = f_d(\boldsymbol{m}, \boldsymbol{y}, \boldsymbol{\theta}_d)$, where $\boldsymbol{x}, \boldsymbol{y}$ are input embeddings, $\boldsymbol{m} = f_e(\boldsymbol{x}, \boldsymbol{\theta}_e)$ is the encoder memory and $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$ are free parameters to be learned. Since input embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$ are also trained together with the model, we use $\boldsymbol{\theta}_E = \{\boldsymbol{x}, \boldsymbol{\theta}_e\}$ and $\boldsymbol{\theta}_D = \{\boldsymbol{y}, \boldsymbol{\theta}_d\}$ to denote the entire sets of trainable parameters for encoder and decoder respectively. Using shorthand notation, the model can then be written as $f(\boldsymbol{\theta}) = f_d(\boldsymbol{m}, \boldsymbol{\theta}_D)$ and $\boldsymbol{m} = f_e(\boldsymbol{\theta}_E)$.

The goal can be formulated as bounding the decoder update:

$$\Delta f_d = f_d\left(\tilde{\boldsymbol{m}}, \boldsymbol{\theta}_D - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_D}\right) - f_d(\boldsymbol{m}, \boldsymbol{\theta}_D) \quad (2)$$

where $\tilde{\boldsymbol{m}} = f_e\left(\boldsymbol{\theta}_E - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_E}\right)$ is the updated memory. Using Taylor expansion we get that the SGD update is proportional to the magnitude of the gradient:

$$\begin{aligned}\Delta f &= \frac{\partial f}{\partial \boldsymbol{\theta}_D}\Delta\boldsymbol{\theta}_D + \frac{\partial f}{\partial \boldsymbol{\theta}_E}\Delta\boldsymbol{\theta}_E + O(||\Delta\boldsymbol{\theta}_E||^2 + ||\Delta\boldsymbol{\theta}_D||^2) \\ &= -\eta\left(\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}^T\frac{\partial \mathcal{L}}{\partial f_d}^T + \frac{\partial f_d}{\partial f_e}\frac{\partial f_e}{\partial \boldsymbol{\theta}_E}\frac{\partial f_e}{\partial \boldsymbol{\theta}_E}^T\frac{\partial f_d}{\partial f_e}^T\frac{\partial \mathcal{L}}{\partial f_d}^T\right) \\ &\quad + O(\eta^2)\end{aligned} \quad (3)$$

Assuming that $\left\|\frac{\partial \mathcal{L}}{\partial f_d}\right\| = \Theta(1)$, this implies that we need to bound the magnitudes of, $\frac{\partial f_e}{\partial \boldsymbol{\theta}_E}$, $\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}$ and $\frac{\partial f_d}{\partial f_e}$ from above and below to achieve the target goal.

We begin with the decoder and its gradient $\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}$. The decoder is a sequence of L_d residual blocks $G_1, \ldots, G_{L_d}$ such that $\boldsymbol{y}_{l+1} = \boldsymbol{y}_l + G_l(\boldsymbol{y}_l; \boldsymbol{\theta}_{dl})$, where $\boldsymbol{y}_l$ and $\boldsymbol{\theta}_{dl}$ are inputs and parameters of the l'th decoder block respectively. Due to the residual architecture of each block with direct skip connections, we have that $\frac{\partial f_d}{\partial \boldsymbol{y}_l}\frac{\partial f_d}{\partial \boldsymbol{y}_l}^T = \Theta(1)$. The gradient of the decoder can be written as:

$$\begin{aligned}\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}^T &= \frac{\partial f_d}{\partial \boldsymbol{y}}\frac{\partial f_d}{\partial \boldsymbol{y}}^T + \sum_{l=1}^{L_d}\frac{\partial G_l}{\partial \boldsymbol{\theta}_{dl}}\frac{\partial G_l}{\partial \boldsymbol{\theta}_{dl}}^T \\ &\quad + O\left(\sum_{l=1}^{L_d}\frac{\partial G_l}{\partial \boldsymbol{\theta}_{dl}}^4\right)\end{aligned} \quad (4)$$

It follows that if we can bound each summand in Equation 4 to be of order $\Theta(1/L_d)$, then $\|\partial f_d/\partial \boldsymbol{\theta}_D\|^2 = \Theta(1)$ and the goal is achieved. Zhang et al. (2019b) showed that $\Theta(1/L_d)$ bound can be obtained for the MLP blocks with appropriate initialization, and we proceed to derive analogous initialization for the attention blocks.

The Transformer attention consists of keys, queries and values that are combined via softmax operation. Given $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c} \in \mathbb{R}^{n\times d}$ representing the input keys, queries and values respectively, let $\boldsymbol{k}, \boldsymbol{q}, \boldsymbol{v} \in \mathbb{R}^{d\times d'}$ denote the projection matrices for each input. Moreover, let $\boldsymbol{w} \in \mathbb{R}^{d\times d'}$ denote the output projection matrix. The attention is then defined as:

$$\text{Attn}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \text{softmax}\left(\frac{1}{\sqrt{d}}\boldsymbol{b}\boldsymbol{q}\boldsymbol{k}^T\boldsymbol{a}^T\right)\boldsymbol{c}\boldsymbol{v}\boldsymbol{w}^T \quad (5)$$

where the softmax operation is applied across the rows. In Transformer two versions of attention are used, the encoder has self-attention where keys, queries and values are the same. The decoder alternates between self-attention and encoder-attention where keys and values are taken from the encoder memory $\boldsymbol{m}$. We focus on the encoder-attention here as analogous derivation follows trivially for the self-attention.

Our aim is to constrain the magnitude of the update for the encoder-attention. This leads to the main result of this work, where we show that with appropriate scaling of projection matrices and encoder memory, we can get the desired bound. Since we are only considering the magnitude of the update it is sufficiently instructive to study the case where $d = d' = 1$. In this case the projection matrices reduce to scalars k, q, v, $w \in \mathbb{R}$, and $\boldsymbol{m}$ is a $n \times 1$ vector. Let $m_i \in \mathbb{R}$ denote the i'th element of $\boldsymbol{m}$. Our main result follows:

Theorem 3.1. *Let* $G_l(\boldsymbol{m}, \boldsymbol{y}_l; \boldsymbol{\theta}_{dl}) = \text{Attn}(\boldsymbol{m}, \boldsymbol{y}_l, \boldsymbol{m})$, *assuming that* $\|\partial\mathcal{L}/\partial G_l\| = \Theta(1)$, *then* $\Delta G_l \triangleq G_l\left(\boldsymbol{m} - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{m}}, \boldsymbol{y}_l; \boldsymbol{\theta}_{dl} - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_{dl}}\right) - G_l(\boldsymbol{m}, \boldsymbol{y}_l; \boldsymbol{\theta}_{dl})$ *satisfies* $\|\Delta G_l\| = \Theta(\eta/L_d)$ *when:*

$$\|v\|^2\|w\|^2 + \|w\|^2\|m_i\|^2 + \|v\|^2\|m_i\|^2 = \Theta(1/L_d)$$

for all $i = 1, \ldots, n$.

The proof of this theorem is detailed in Appendix A.

Applying Theorem 3.1 to self-attention blocks in the encoder, we get that $\frac{\partial f_d}{\partial \boldsymbol{\theta}_E}$ is bounded if the following condition is satisfied:

$$L_e\left(\|v_e\|^2\|x\|^2 + \|w_e\|^2\|x\|^2 + \|v_e\|^2\|w_e\|^2\right) = \Theta(1) \quad (6)$$

where v_{el} and w_{el} are parameters in the l'th decoder block with $v_{el} = \Theta(v_e)$ and $w_{el} = \Theta(w_e)$ for all l. Following similar derivation for the decoder, we get that $\frac{\partial f_d}{\partial \boldsymbol{\theta}_D}$ is bounded if:

$$\begin{aligned}&L_d(\|v_d\|^2\|w_d\|^2 + \|v_d\|^2\|y\|^2 + \|w_d\|^2\|y\|^2 \\ &\quad + \|v_d\|^2\|w_d\|^2 + \|v_d\|^2\|m\|^2 + \|w_d\|^2\|m\|^2) = \Theta(1)\end{aligned} \quad (7)$$

where L_d is the number of blocks in the decoder, v_{dl} and w_{dl} are parameters in the l'th block with $v_{dl} = \Theta(v_d)$ and $w_{dl} = \Theta(w_d)$ for all l. Finally, using the fact that encoder and decoder are connected via memory in the encoder-attention blocks, we have that $\frac{\partial f_d}{\partial f_e} = \frac{\partial f_d}{\boldsymbol{m}}$ is bounded if:

$$L_d(\|v_d\|^2\|w_d\|^2) = \Theta(1) \quad (8)$$

Derivations for these conditions are outlined in Appendix B.

By appropriately initialising weights in each block of encoder and decoder, we can ensure that the above conditions are satisfied during the initial updates. As we have

Model	IWSLT'14$_{small}$ De-En	IWSLT'14$_{small}$ En-De	WMT'18$_{base}$ Fi-En	WMT'17$_{base}$ En-De	WMT'17$_{big}$ En-De
Baseline	34.2 (Zhang et al., 2019b)	28.6 Liu et al. (2020)	25.25 (Rikters, 2018)	27.3 (Vaswani et al., 2017)	29.3 (Ott et al., 2018)
Pre-LN (Chen et al., 2018b)	–	–	–	27.1	28.7
R-Fixup (Zhang et al., 2019b)	34.5	–	–	–	29.3
RAdam, no warmup (Liu et al., 2020)	34.8	28.5	–	–	–
T-Fixup	**35.5**	**29.4**	**25.7**	**29.1**	**29.7**

Table 1. Test BLEU scores on all datasets. We benchmark *small* models for IWSLT'14, *base* models for WMT'17 and WMT'18, and *large* models for WMT'17. Unless indicated all baselines use warmup and layer normalization, T-Fixup removes both.

shown, early updates can be particularly unstable so bounding their magnitude can prevent optimization from diverging. We empirically demonstrate that this is indeed the case, and further show that it allows to remove both LN and warmup. There are multiple initialisation schemes that would satisfy the target conditions. Given that an N-layer Transformer has $L_e = 2N$ and $L_d = 3N$ blocks in the encoder and decoder respectively, in this work we use $\|v_d\| = \|w_d\| = \|\boldsymbol{x}\| = \|\boldsymbol{y}\| = (3L_d)^{-\frac{1}{4}} = (9N)^{-\frac{1}{4}}$. This satisfies Equations 7 and 8, and we then solve for Equation 6. Assuming $\|v_e\| = \|w_e\|$ due to symmetry, we obtain $\|v_e\| = \|w_e\| \approx 0.67N^{-\frac{1}{4}}$ (see Appendix C). We then apply our initialisation scheme as follows:

- Apply Xavier initialization for all parameters excluding input embeddings. Use Gaussian initialization $\mathcal{N}(0, d^{-\frac{1}{2}})$ for input embeddings where d is the embedding dimension.
- Scale $\boldsymbol{v}_d$ and $\boldsymbol{w}_d$ matrices in each decoder attention block, weight matrices in each decoder MLP block and input embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$ in encoder and decoder by $(9N)^{-\frac{1}{4}}$.
- Scale $\boldsymbol{v}_e$ and $\boldsymbol{w}_e$ matrices in each encoder attention block and weight matrices in each encoder MLP block by $0.67N^{-\frac{1}{4}}$.

After initialisation, we remove layer normalization from all blocks and train without learning rate warmup. We refer to this initialization scheme as T-Fixup. To the best of our knowledge the closest related work is that of Zhang et al. (2019b) that proposed initialization for residual MLP blocks (ResNet models); we refer to that approach as R-Fixup. We note that in addition to parameter scaling, R-Fixup applies modifications to the model architecture such as adding scaling and bias layers to each block. The Depth-Scaled Initialization (Zhang et al., 2019a) is also inspired by R-Fixup, but to avoid adding extra layers the authors keep layer normalization, and instead modify attention structure to improve performance. Our approach, in contrast, allows to simply remove layer normalization without any other modifications to the model. We demonstrate in the experiments section that by appropriately re-scaling the weight matrices, we can train arbitrarily deep Transformer models with leading performance.

4. Experiments

We compare Transformer trained with our initialization against leading models on multiple public NMT benchmarks including IWSLT'14 De-En, WMT'17 En-De and low resource language pair WMT'18 Fi-En. We compare our approach against the original baseline model (Vaswani et al., 2017), and leading recently proposed Transformer optimization methods including Pre-LN (Chen et al., 2018b), DLCL (Wang et al., 2019), R-Fixup (Zhang et al., 2019b), RAdam (Liu et al., 2020) and DS-Init (Zhang et al., 2019a)

All experiments are done using the Fairseq library (Gehring et al., 2017). To stay consistent with previous work we train three model sizes: *small* 512-1024-4, *base* 512-2048-8 and *big* 1024-4096-16; where the numbers correspond to embedding dimension, MLP layer size and number of attention heads respectively. All models have 6 layers in both encoder and decoder, and we investigate deeper models in Section 4.2. To accelerate experiments *big* and *base* models on the WMT datasets are trained with mixed precision. Hyper-parameters for each model are chosen through grid search and are listed in Appendix D. To demonstrate that our initialization works well with the Adam optimizer we use Adam for all experiments. Training is done on an IBM server with 160 POWER9 CPUs, 600GB RAM and 4 Tesla V100 GPUs. We use test BLEU metric to compare model performance.

4.1. NMT Results

The NMT results are shown in Table 1. From the table we see that T-Fixup performs comparably or beats all baselines on each dataset. The performance is particularly strong on

Model	Layers	BLEU
Baseline	6	27.3
Pre-LN Chen et al. (2018b)	20	28.9
DLCL (Wang et al., 2019)	25	29.2
DLCL-Pre-LN (Wang et al., 2019)	30	29.3
T-Fixup	6	29.1
	20	29.4
	30	**29.7**

Table 2. WMT'17 En-De test BLEU results for different versions of the Transformer base architecture with varying depth.

Model	Layers	BLEU
Baseline	6	27.6
DS-Init (Zhang et al., 2019a)	12	28.6
	20	28.7
LRI (Xu et al., 2019)	12	28.7
	24	29.5
T-Fixup	12	29.3
	20	29.6
	30	**30.1**

Table 3. WMT'14 En-De test BLEU results for different versions of the Transformer base architecture with varying depth. T-Fixup models were trained with the same parameter settings as in Zhang et al. (2019a)

the WMT'17 En-De dataset where our base model outperforms the best baseline by nearly 2 BLEU points. These results indicate that by using our simple initialization scheme we can effectively remove both layer normalization and warmup, and still achieve leading performance.

We also see that the Pre-LN version tends to hurt performance, particularly for the big model, with over half point drop in BLEU on the WMT'17 En-De dataset. The drop in performance was also noted by Nguyen & Salazar (2019) and Wang et al. (2019), and suggests that moving layer normalization inside each block is not an optimal solution even though it helps with optimization. Finally, R-Fixup performs comparably to the baseline model with a small improvement on the IWSLT'14 De-En dataset. However, as we noted above, R-Fixup requires additional modifications to the architecture, such as adding scaling/bias layers and zeroing-out some weights in each block. Our approach, on the other hand, doesn't require any modifications and achieves better performance. So tailoring weight initialization to the specifics of the Transformer architecture is beneficial.

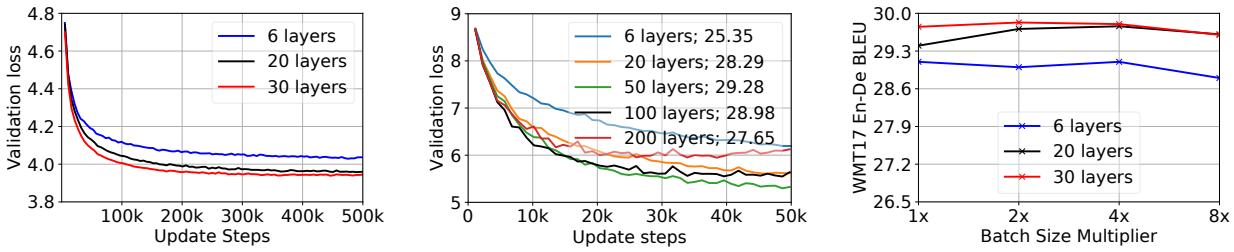
Figures 2(e) and 2(f) show log histograms for gradients and Adam updates for T-Fixup *small* model on the IWSLT'14 De-En datset. Due to higher learning rate under the no-warmup setting, Adam updates are also initially larger. However, since layer normalization is removed, large updates no longer lead to vanishing gradients that destabilize the optimizer (see Figures 2(c) and 2(d) for comparison). Moreover, Figure 2(e) shows that without layer normalization our initialization scheme still keeps gradients stable throughout training. This is in contrast to the baseline model, where removing layer normalization makes training diverge even when warmup is used as shown in Figure 3(c).

4.2. Deep Models

The general trend in Transformer models is that increasing size and depth generally leads to better results if sufficient training data is available (Lan et al., 2020). However, training deeper models has proven to be challenging and many of the published architectures for neural machine translation have fewer than 10 layers. Recently, Wang et al. (2019) showed that encoders with up to 30 layers can be trained with Pre-LN. The authors also proposed a dynamic linear combination of layers (DLCL) approach where additional weighted residual connections are introduced between Transformer blocks to improve information flow. DLCL eliminates the need for Pre-LN and improves optimization for deep models. All of these methods, however, still use warmup and some form of layer normalization. We thus investigate whether T-Fixup can be used to successfully train deep models without either of these requirements.

Table 2 shows WMT'17 En-De test BLEU results for various versions of the Transformer base architecture with varying depth. We compare against deep versions of Pre-LN and DLCL models with 20 and 25 layers respectively. For DLCL, we also compare against the Pre-LN version DLCL-Pre-LN with 30 layers. We train deeper versions of our model by simply replicating layers, and no additional structural modifications are done.

From Table 2 we see that increasing depth does improve performance as all deep models outperform the 6 layer baseline by over 1 point in BLEU. The DLCL has small improvement over the Pre-LN approach, although the comparison is not entirely fair as Pre-LN has fewer layers. We also see that our approach outperforms all baselines of comparable depth. Notably, T-Fixup with 6 layers performs comparably to Pre-LN and DLCL baselines with 20 and 25 layers respectively. Moreover, T-Fixup with 20 layers outperforms the best baseline DLCL-Pre-LN that has 30 layers. Similar results can be observed from Table 3 where we further compare against two other published baselines that train deep models, DS-Init (Zhang et al., 2019a) and LRI (Xu et al., 2019) on the WMT'14 En-De dataset. We again see that T-Fixup outperforms all baselines of comparable depth. Together these comparisons cover the majority of published results to-date with deep Transformer encoder-decoder models, and demonstrate that T-Fixup can be effectively used to train deep models.



(a) Deep WMT Models (b) Deep IWSLT Models (c) Large Batch Size

Figure 4. 4(a) shows validation curves on the WMT'17 En-De Dataset for the T-Fixup *base* models of varying depth. 4(b) shows IWSLT'14 De-En validation curves for the T-Fixup model with a 64-128-2 configuration as depth is increased from 6 to 200 layers in both encoder and decoder. Test BLEU results are also shown in the legend for each model. 4(c) shows test BLEU results on the WMT'17 En-De dataset for the T-Fixup *base* model as batch size is increased by a factor of 2x to 8x. All other parameters are kept fixed.

Figure 4(a) shows WMT'17 En-De validation curves for T-Fixup *base* models as depth is varied from 6 to 30 layers. We see that the validation curves remain stable throughout learning even up to 500K updates. This indicates that with appropriate initialization, a much higher initial learning rate (5e-4 instead of 1e-7) can be used without destabilizing training. T-Fixup of the same depth as the baseline trains faster, and reaches better validation loss which translates to better test BLEU as shown in Table 2. The benefit of increasing depth is also clearly observed from this figure, where 20 and 30 layer models achieve better validation losses for most of the training period.

To further investigate whether we can successfully train very deep models, we significantly increase the depth of the model to 200 layers in both encoder and decoder. The model thus has 400 layers in total and over 1000 attention/MLP blocks. To fit model of this depth on a GPU we reduce the size to 64-128-2 corresponding to embedding dimension, MLP layer size and number of attention heads respectively. As before, we simply apply T-Fixup to initialize the weights and remove warmup and layer normalization; no other modifications are done.

Figure 4(b) shows IWSLT'14 De-En validation curves for the first 50K updates and test BLEU scores for the T-Fixup models as depth is increased from 6 to 200 layers. From the figure we see that all models successfully train and have smooth validation curves. After 50 layers we observe over-fitting which can be attributed to the small dataset size. However, all models achieve reasonable test BLEU scores, particularly given their reduced block size and short training time, and we see no indication of divergence or other optimization problems. These results further support the conclusion that our initialization can be successfully used to train very deep Transformer models opening up avenues for further research in this area.

It has been shown that learning rate warmup is important for large batch training in Transformer models. The general finding is that longer warmup is needed for larger batch sizes (Popel & Bojar, 2018). Since T-Fixup removes the warmup phase, we test how it performs with large batch training. Figure 4(c) shows test BLEU scores on the WMT'17 En-De datasets for the T-Fixup *base* model as batch size is increased by a factor of 2x to 8x. In this set-up the largest batch size at 8x has 200K tokens. For each experiment we only change the batch size and leave all other parameters fixed. From Figure 4(c) we see that all models from 6 to 30 layers train without problems even on largest batch size setting. During training we further observe that convergence speed scales proportionally to batch size multiplier making multi-GPU training highly effective. These results indicate that proper initialisation can also successfully replace warmup for large batch training. The BLEU performance effect from increasing batch size is generally negligible, with all models performing comparably across different batch sizes. We suspect that this is due to the fact that all other parameters are kept fixed, and better results can be obtained by further tuning them.

5. Conclusion

We present an investigation into optimization challenges in the Transformer encoder-decoder models. Specifically, we first show that the requirement for learning rate warmup comes from the instability in the Adam optimizer combined with gradient vanishing through layer normalization. We then propose a new weight initialization scheme with theoretical guarantees that keeps model updates bounded, and allows to remove both warmup and layer normalization. Experiments on public NMT benchmarks shows that we can successfully train very deep models with over 1000 MLP/attention blocks and achieve leading results. Future work involves further investigation into Adam and other commonly used optimizers, as well as deriving analogous initialization schemes for other architectures.

Acknowledgements

This work is funded by the Mitacs Accelerate program. We would like to thank Vector Institute and University of Toronto's Department of Computer Science for their support on this project. We would also like to give a special thanks to Roberto Soares, for setting up and configuring the servers for the large scale experiments.

References

Ahmed, K., Keskar, N. S., and Socher, R. Weighted transformer network for machine translation. In *arXiv:1711.02132*, 2017.

Aji, A. F. and Heafield, K. Making asynchronous stochastic gradient descent work for transformers. In *ACL: Neural Generation and Translation*, 2019.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. In *arXiv:2004.05150*, 2020.

Bilkhu, M., Wang, S., and Dobhal, T. Attention is all you need for videos: Self-attention based video summarization using universal transformers. In *arXiv:1906.02792*, 2019.

Chen, M., Li, Y., Zhang, Z., and Huang, S. TVT: Two-View transformer network for video captioning. In *ACML*, 2018a.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*, 2018b.

Chen, Q., Zhao, H., Li, W., Huang, P., and Ou, W. Behavior sequence transformer for e-commerce recommendation in Alibaba. *KDD*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. In *EMNLP*, 2018.

Fan, L., Huang, W., Gan, C., Ermon, S., Gong, B., and Huang, J. End-to-end learning of motion representation for video understanding. In *CVPR*, 2018.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *ICML*, 2017.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A Lite BERT for self-supervised learning of language representations. In *ICLR*, 2020.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020.

Ma, J. and Yarats, D. On the adequacy of untuned warmup for adaptive optimization. In *arXiv:1910.04209*, 2019.

Nguyen, T. and Salazar, J. Transformers without tears: Improving the normalization of self-attention. In *arXiv:1910.05895*, 2019.

Ott, M., Edunov, S., Grangier, D., and Auli, M. Scaling neural machine translation. In *ACL*, 2018.

Popel, M. and Bojar, O. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110:43–70, April 2018.

Rikters, M. Impact of corpora quality on neural machine translation. In *arXiv:1810.08392*, 2018.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D., and Chao, L. Learning deep transformer models for machine translation. In *ACL*, 2019.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture. In *arXiv:2002.04745*, 2020.

Xu, H., Liu, Q., van Genabith, J., and Zhang, J. Why deep transformers are difficult to converge? From computation order to Lipschitz restricted parameter initialization. In *arXiv:1911.03179*, 2019.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Zhang, B., Titov, I., and Sennrich, R. Improving deep transformer with depth-scaled initialization and merged attention. In *EMNLP*, 2019a.

Zhang, H., Dauphin, Y. N., and Ma, T. Residual learning without normalization via better initialization. In *ICLR*, 2019b.