ReZero is All You Need: Fast Convergence at Large Depth

Thomas Bachlechner[*], **Bodhisattwa Prasad Majumder**[*], **Huanru Henry Mao**[*]
Garrison W. Cottrell, Julian McAuley
UC San Diego
{tbachlechner@physics, bmajumde@eng, hhmao@eng, gary@eng, jmcauley@eng}
.ucsd.edu

Abstract

Deep networks have enabled significant performance gains across domains, but they often suffer from vanishing/exploding gradients. This is especially true for Transformer architectures where depth beyond 12 layers is difficult to train without large datasets and computational budgets. In general, we find that inefficient signal propagation impedes learning in deep networks. In Transformers, multi-head self-attention is the main cause of this poor signal propagation. To facilitate deep signal propagation, we propose ReZero, a simple change to the architecture that initializes an arbitrary layer as the identity map, using a single additional learned parameter per layer. We apply this technique to language modeling and find that we can easily train ReZero-Transformer networks over a hundred layers. When applied to 12 layer Transformers, ReZero converges 56% faster on enwiki8. ReZero applies beyond Transformers to other residual networks, enabling 1,500% faster convergence for deep fully connected networks and 32% faster convergence for a ResNet-56 trained on CIFAR 10.

1 Introduction

Deep learning has enabled significant improvements in state-of-the-art performance across domains [1, 2, 3, 4]. The expressivity of neural networks typically grows exponentially with depth [5], enabling strong generalization performance, but often induces vanishing/exploding gradients and poor signal propagation through the model [6]. Researchers have relied on careful initialization [7, 8] and normalization techniques such as BatchNorm [9] and LayerNorm [10] to mitigate this issue, but these techniques can be costly and limited.

In this work, we propose ReZero[2], a small architectural addition that dynamically facilitates well-behaved gradients and arbitrarily deep signal propagation. The idea is simple: ReZero initializes each layer to perform the identity operation. For each layer, we introduce a residual connection for the input signal $\boldsymbol{x}$ and one trainable parameter α that modulates the non-trivial transformation of the layer $F(\boldsymbol{x})$,

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F(\boldsymbol{x}_i)\,, \tag{1}$$

where $\alpha = 0$ at the beginning of training. Initially the gradients for all parameters defining F vanish, but dynamically evolve to suitable values during initial stages of training. We illustrate the architecture in Figure 1.

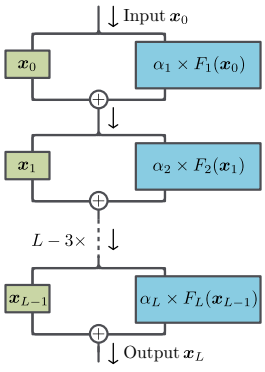

Figure 1: ReZero

[*]Authors contributed equally, ordered by last name

[2]Code for ReZero applied to various neural architectures: https://github.com/majumderb/rezero

Preprint. Work in progress.

Table 1: Various forms of normalization and residual connections. F represents the transformation of an arbitrary layer and "Norm" is a normalization (e.g., LayerNorm or BatchNorm).

(1) Deep Network	$\boldsymbol{x}_{i+1} = F(\boldsymbol{x}_i)$
(2) Residual Network	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\boldsymbol{x}_i)$
(3) Deep Network + Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(F(\boldsymbol{x}_i)\big)$
(4) Residual Network + Pre-Norm	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\text{Norm}\big(\boldsymbol{x}_i\big))$
(5) Residual Network + Post-Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(\boldsymbol{x}_i + F(\boldsymbol{x}_i)\big)$
(6) **ReZero**	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F(\boldsymbol{x}_i)$

ReZero provides two main benefits:

Deeper learning — Signals effectively propagate through deep networks, which allows for learning in otherwise untrainable networks. ReZero successfully trains 10,000 layers of fully-connected networks, and we are the first to train Transformers over 100 layers without learning rate warm-up or LayerNorm. In contrast to [11] we find that to get good results at this depth, it is not necessary to add auxiliary losses.

Faster convergence — We observe significantly accelerated convergence in ReZero networks compared to regular residual networks with normalization. When ReZero is applied to Transformers, we converge 56% faster than the vanilla Transformer to reach 1.2 BPB on the enwiki8 language modeling benchmark. When applied to ResNets, we obtain 32% speed up to reach 85% accuracy on CIFAR 10.

2 Background and related work

Networks with a depth of L layers and width w often have an expressive power that scales exponentially in depth, but not in width [12, 5]. Large depth often comes with difficulty in training via gradient-based methods. During training of a deep model, a signal in the training data has to propagate forward from the input to the output layer, and subsequently, the cost function gradients have to propagate backwards in order to provide a meaningful weight update. If the magnitude of a perturbation is changed by a factor r in each layer, both signals and gradients vanish or explode at a rate of r^L, rendering many deep networks untrainable in practice.

To be specific, consider a deep network that propagates an input signal $\boldsymbol{x}_0$ of width w through L layers that perform the non-trivial, but width preserving functions $F[\mathcal{W}_i] : \mathbb{R}^w \to \mathbb{R}^w$, where $\mathcal{W}_i$ denotes all parameters at layer $i = 1, \ldots, L$. The signal propagates through the network according to

$$\boldsymbol{x}_{i+1} = F[\mathcal{W}_i](\boldsymbol{x}_i)\,. \tag{2}$$

There have been many attempts to improve signal propagation through deep networks, and they often fall into one of three categories — initialization schemes, normalization layers, and residual connections. We show some of the popular ways to combine residual networks with normalization in Table 1.

2.1 Careful initialization

In recent years the dynamics of signal propagation in randomly initialized deep and wide neural networks have been formalized via mean field theory [13, 8, 14]. For some deep neural networks, including fully connected and convolutional architectures, the cosine distance of two distinct signals, $\boldsymbol{x}_i \cdot \boldsymbol{x}'_i / (\|\boldsymbol{x}_i\| \|\boldsymbol{x}'_i\|)$, approaches a fixed point that either vanishes or approaches unity at large depths. If this fixed point is 1 the behavior of the network is stable and every input is mapped to the same output, leading to vanishing weight updates. If this fixed point is 0 the behavior of the network is chaotic and even similar inputs are mapped to very different outputs, leading to exploding weight updates. To understand whether a network is in a stable or chaotic phase we consider the input-output Jacobian

$$\boldsymbol{J}_{\text{io}} \equiv \frac{\partial \boldsymbol{x}_L}{\partial \boldsymbol{x}_0}\,. \tag{3}$$

The mean squared singular values χ of this matrix determine the growth/decay of an average input signal perturbation as it propagates through the network. The network exhibits a boundary between

the ordered and the chaotic phase, the edge of chaos at $\chi = 1$. Training proceeds efficiently at the edge of chaos.

This behavior was recognized in [15, 6], which motivated a re-scaling of the weights such that $\chi \approx 1$ and on average signal strengths are neither enhanced or attenuated.

Pennington et al. [13, 14] recognized that a unit mean squared average of the input-output Jacobian is insufficient to guarantee trainability. For example, if the singular vectors of $\boldsymbol{J}_{\text{io}}$ corresponding to very large/small singular values align well with the perturbations in the data, training will still be inefficient. They proposed the stronger condition of *dynamical isometry* [16], which requires that all singular values of $\boldsymbol{J}_{\text{io}}$ are close to one. This means that all perturbations of the input signal propagate through the network equally well. The ReLU activation function maps to zero for some perturbations of the input signal, and it is therefore intuitive that deep networks with ReLU activations cannot possibly satisfy dynamical isometry, as was rigorously established in [13]. For some activation functions and network architectures, elaborate initialization schemes allow the network to satisfy dynamical isometry at initialization, which significantly improves training dynamics [17, 5, 18, 19].

2.2 Normalization

An alternative approach to improve the trainability of deep neural networks is to incorporate layers that explicitly provide normalization. Many normalization modules have been proposed, with the two most popular ones being BatchNorm [9] and LayerNorm [10]. In general, normalization aims to ensure that initially, signals have zero mean and unit variance as they propagate through a network, reducing "covariate shift" [9]. For simplicity we will focus primarily on comparisons against LayerNorm because BatchNorm has additional regularizing effects that are orthogonal to our investigation.

Normalization methods have shown success in accelerating the training of deep networks, but they do incur a computational cost to the network and pose additional hyperparameters to tune (e.g., where to place the normalization). In contrast to normalization methods, our proposed method is simple and cheap to implement. ReZero alone is sufficient to train deeper networks, even in the absence of various norms. Although ReZero makes normalization superfluous for convergence, we have found the regularizing effect of BatchNorm to be complementary to our approach.

2.3 Residual connections

The identity mappings introduced in [2] enabled a deep residual learning framework in the context of convolutional networks for image recognition that significantly increased the trainable depth. The complementary use of BatchNorm and ResNets [2] has enabled the training of convolutional neural networks with over 100 layers. The same has not been the case for LayerNorm and Transformer architectures. Yang et al. [18] studied residual fully connected networks and demonstrated that due to the skip connection, signals decay more slowly (polynomially) as they propagate, allowing for effective training of deeper networks.

Concurrently with our work SkipInit [20], an alternative to the BatchNorm, was proposed for ResNet architectures that is similar to ReZero. The authors find that in deep ResNets without BatchNorm, a scalar multiplier is needed to ensure convergence. We arrive at a similar conclusion for the specific case considered in [20], and study more generally signal propagation in deeper networks across multiple architectures and beyond BatchNorm.

3 ReZero

We propose **ReZero** (**re**sidual with **zero** initialization), a simple change to the architecture of deep residual networks that facilitates dynamical isometry and enables the efficient training of extremely deep networks. Rather than propagating the signal through each of the non-trivial functions $F[\mathcal{W}_i]$ at initialization, we add a skip connection and rescale the function by L learnable parameters α_i (which we call *residual weights*) that are initialized to zero. The signal now propagates according to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F[\mathcal{W}_i](\boldsymbol{x}_i)\,. \tag{4}$$

At initialization the network represents the identity function and it trivially satisfies dynamical isometry. We demonstrate below for a toy model that this architecture can exponentially accelerate

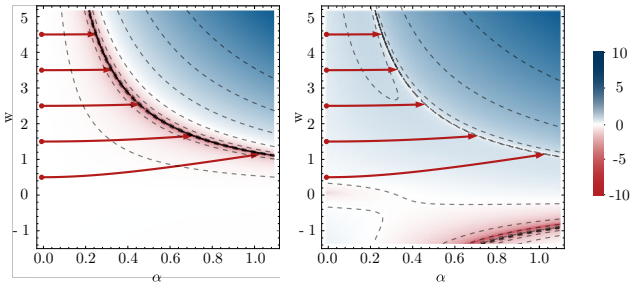

Figure 2: Contour log plots of a quadratic cost function (left) and gradient norm (right) over the network weight w and the residual weight α during the training of the linear function $x_L = 5x_0$ via gradient descent using a training set of $x_0 = \{1, 2, 3\}$. Gradient descent trajectories initialized at $\alpha = 0$ are shown in red for five different initial w's. The trajectory dynamics avoid the poorly conditioned regions around $\alpha \approx 1$.

training. The architecture modification allows for the training of deep networks even when the individual layers' Jacobian has vanishing singular values, as is the case for ReLU activation functions or self-attention [21]. The technique also allows us to add arbitrary new layers to existing and trained networks.

3.1 A toy example

To illustrate how the ReZero connection accelerates training let us consider the toy model of a deep neural network described by L single-neuron hidden layers that have no bias and all share the same weight w and $\alpha_i = \alpha \; \forall i$. The network then simply maps an input x_0 to the output

$$x_L = (1 + \alpha w)^L x_0 \, . \tag{5}$$

Fixing the parameter $\alpha = 1$ would represent a toy model for a fully connected residual network, while initializing $\alpha = 0$ and treating α as a learned parameter corresponds to a ReZero network. The input-output Jacobian is given by $J_{\text{io}} = (1 + \alpha w)^L$, indicating that for initialization with $w \approx 1$ and $\alpha = 1$ the output signal of a deep (i.e., $L \gg 1$) network is extremely sensitive to any small perturbations of the input, while with $\alpha = 0$ the input signal magnitude is preserved. While this example is too simple to exhibit an order/chaos phase transition, it does accurately model the vanishing and exploding gradient problem familiar in deep networks. Assuming a learning rate λ and a cost function $\mathcal{C}$, gradient descent updates the weights w according to

$$w \leftarrow w - \lambda L \alpha x_0 (1 + \alpha w)^{L-1} \partial_x \mathcal{C}(x)|_{x=x_L} \, . \tag{6}$$

For $\alpha = 1$, convergence of gradient descent with an initial weight $w \approx 1$ requires steps no larger than 1, and hence a learning rate that is exponentially small in depth L

$$\lambda \propto L^{-1} (1 + w)^{-(L-1)} \, , \tag{7}$$

where we only retained the parametric dependence on w and L. For $w \gg 1$ the gradients in Equation 6 explode, while for $w \approx -1$ the gradients vanish. Initializing $\alpha = 0$ solves both of these problems: assuming a sufficiently well-conditioned cost function, the first step of gradient descent will update the residual weights α to a value that avoids large outputs and keeps the parameter trajectory within a well-conditioned region while retaining the expressive power of the network. The first non-trivial steps of the residual weight are given by

$$\alpha \leftarrow -\lambda L w x_0 \partial_x \mathcal{C}(x)|_{x=x_L} \, , \tag{8}$$

and gradient descent will converge with a learning rate that is polynomial in the depth L of the network. In this simple example, the ReZero connection, therefore, allows for convergence with dramatically fewer optimization steps than a vanilla residual network. We illustrate the training dynamics, cost function and gradients in Figure 2.

4 Training deep fully connected networks faster

We now study the effect of ReZero on deep ReLU networks, and when combined with various approaches that facilitate deep learning listed in the rows of Table 1. Specifically, we will compare a vanilla deep fully connected network (FC, row 1), a deep network with residual connections (FC+Res, row 2) a deep network with LayerNorm (FC+Norm, row 3), and finally our proposal ReZero (row 6). We choose the initial weights $\boldsymbol{W}_i$ normally distributed with variances optimal for training, i.e. $\sigma_W^2 = 2/w$ for all but the vanilla residual network where $\sigma_W^2 \approx 0.25/w$, see [6, 18].

As a sample toy task, we train four different network architectures on the CIFAR-10 data set for supervised image classification. We are only interested in the training dynamics and investigate how many iterations it takes for the model to fit the data.

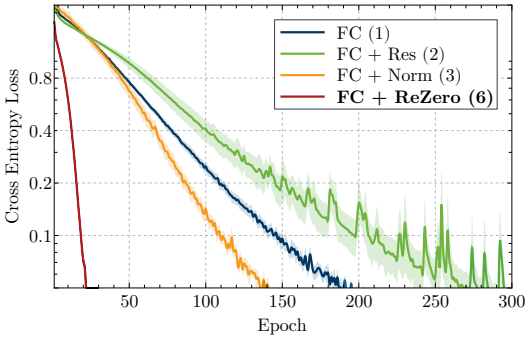

Figure 3: Cross entropy loss during training of four variants of 32 layer fully connected networks with width 256 and ReLU activations. The bracketed numbers refer to the architectures in the corresponding rows of Table 1. We average over five runs each and show 1σ error bands. For all models we use the Adagrad [22] optimizer with learning rate 0.01.

We show the evolution of the training loss in Figure 3. In our simple experiment, a 32 layer network the ReZero architecture converges to fit the training data between 7 and 15 times faster than the other techniques. Note that without an additional normalization layer the residual connection decreases convergence speed compared to a plain fully connected network. We speculate that this is because at initialization the variance of the signal is not independent of depth, see [18].

With increasing depth, the advantages of the ReZero architecture become more apparent. To verify that this architecture ensures trainability to large depths we successfully trained fully connected ReZero networks with up to $10,000$ layers on a laptop with one GPU[3] to overfit the training set.

5 Training deeper Transformers faster

In this section, we study the signal propagation and application of ReZero to the Transformer architecture [21]. Transformers gained significant popularity and success both in supervised and unsupervised NLP tasks [23, 11]. Transformers are built by stacking modules that first perform self-attention, then a point-wise feed-forward transformation.

The original Transformer [21] implementation can be seen as a residual network with post-normalization (row 5 in Table 1). Inside a Transformer module the output of each sublayer is added via a residual connection and then normalized by LayerNorm,

$$\boldsymbol{x}_{i+1} = \text{LayerNorm}\left(\boldsymbol{x}_i + \text{sublayer}(\boldsymbol{x}_i)\right) , \quad (9)$$

where sublayer $\in \{\text{self-attention}, \text{feed-forward}\}$, as illustrated in the left panel of Figure 4.

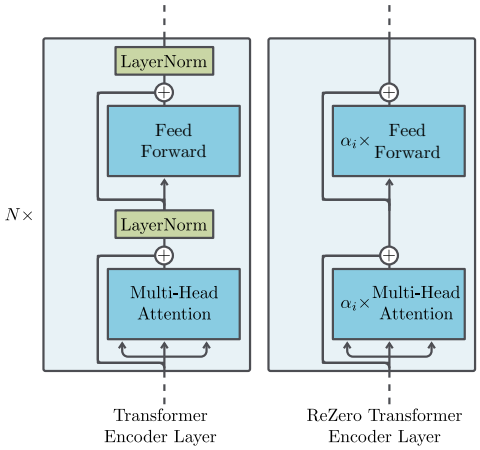

Figure 4: ReZero for Transformers

5.1 Signal propagation in Transformers

Two crucial components relevant to the signal propagation in the original Transformer layers include LayerNorm [10] and (multi-head) self attention [21]. We will argue that neither component by itself or in conjunction with a vanilla residual connection can satisfy dynamical isometry for all input

[3] To train at these extreme depths we used the Adagrad optimizer with a learning rate of 0.003.

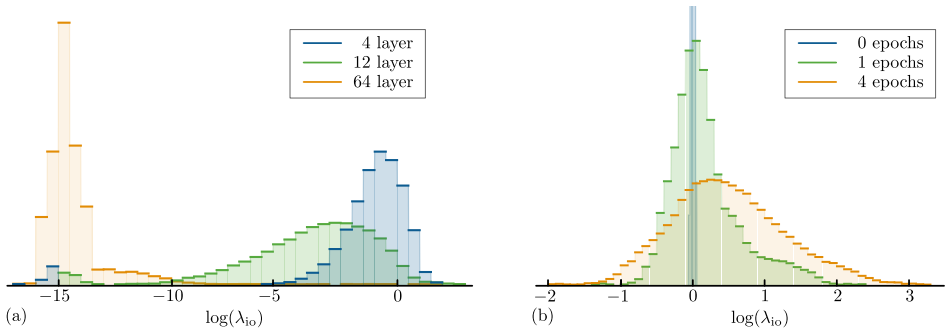

Figure 5: Histograms for log singular values λ_{io} of the input-output Jacobian matrix for: (a) Transformer encoder network at initialization of depths 4, 12 and 64 layers; (b) ReZero Transformer encoder network with 64 layers before and during training. Deep Transformers are far from dynamical isometry, $\lambda_{\text{io}} \ll 1$, while ReZero Transformers remain closer to dynamical isometry with mean singular value $\lambda_{\text{io}} \approx 1$.

signals. This finding motivates the use of a ReZero connection to replace both LayerNorm and the vanilla residual connection.

Layer normalization removes the mean and scales the variance over all neurons of a given layer and introduces learnable parameters γ and β to re-scale the variance and shift the mean according to

$$\text{LayerNorm}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \text{E}(\boldsymbol{x})}{\sqrt{\text{Var}(x)}} \times \gamma + \beta \,. \tag{10}$$

It is clear from this definition that perturbing an input x by a transformation that purely shifts either its mean or variance will leave the output unchanged. These perturbations, therefore, give rise to two vanishing singular values of the input-output Jacobian. In the Transformer architecture [21] the norm is applied to each of the n elements of the input sentence, leading to a total of $2 \times n$ vanishing singular values of the Jacobian for each Transformer layer.

Self-attention allows the model to relate content located across different positions by computing a weighted sum of an input sequence. Specifically, the $n \times d$ matrix $\boldsymbol{x}$ contains an input sequence of n rows containing d-dimensional embedding vectors, from which we can evaluate the query, key and value matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} = \boldsymbol{x} \cdot \boldsymbol{W}^{Q,K,V}$, where the $W^{Q,K,V}$ matrices are $d \times d$. The scaled dot-product attention then is given by

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\boldsymbol{Q} \cdot \boldsymbol{K}^{\top} / \sqrt{d}\right) \cdot \boldsymbol{V} \,. \tag{11}$$

In general, the singular value spectrum of the Jacobian of this attention process is complicated. Rather than studying it in full generality, we now merely argue that for some inputs $\boldsymbol{x}$ and weights $\boldsymbol{W}^{Q,K,V}$ the Jacobian has a large number of vanishing singular values (a claim we evaluate empirically below). Consider weights or inputs such that each of the arguments of the softmax function is small compared to 1. The softmax function then simply returns a $n \times n$ dimensional matrix filled with entries that all approximate $1/n$. This means that the attention function projects all embedding vectors of the input sequence onto a single diagonal direction. This implies that out of the $n \times d$ Jacobian singular values only d are non-vanishing and hence much of the input signal is lost. A residual connection can restore some of the lost signals, but even then some perturbations are amplified while others are attenuated. This example demonstrates that self-attention is incompatible with dynamical isometry and unimpeded signal propagation in deep Transformer networks. It is easy to verify that the same conclusion holds for multi-head attention. A careful initialization of the weights might alleviate some of these issues, but we are not aware of any initialization scheme that would render a Transformer layer consistent with dynamical isometry.

We gave a theoretical argument that the vanilla Transformer contains elements that inhibit deep signal propagation. Here, we verify these claims in practice by obtaining the input-output Jacobian for the attention process by evaluating its change under an infinitesimal variation of each of the $n \times d$ entries of the input sequence $\boldsymbol{x}$. We show the input-output Jacobian for Transformer encoder layers of various depth with Xavier uniform initialized weights in Figure 5a. While shallow Transformers exhibit a singular value distribution peaked around unity, we clearly observe that the Jacobian of

Table 2: Comparison of various 12 layer Transformers normalization variants against ReZero and the training iterations required to reach 1.2 BPB on `enwiki8` validation set.

Model	Iterations	Speedup
Post-Norm [21]	Diverged	-
+ Warm-up	13,690	1×
Pre-Norm	17,765	0.77×
GPT2-Norm [4]	21,187	0.65×
ReZero $\alpha = 1$	14,506	0.94×
ReZero $\boldsymbol{\alpha = 0}$	**8,800**	**1.56**×

Table 3: Comparison of Transformers (TX) on the `enwiki8` test set.

Model	Parameters	BPB
Character TX 12L [11]	41M	1.11
TX 12L + Warm-up	38M	1.17
TX 12L + ReZero $\alpha = 1$	34M	1.17
TX 12L + ReZero $\alpha = 0$	34M	1.17
Character TX 64L [11]	219M	1.06
TX 64L	51M	Diverged
TX 64L + Warm-up	51M	Diverged
TX 64L + ReZero $\alpha = 1$	51M	Diverged
TX 64L + ReZero $\alpha = 0$	51M	1.11
TX 128L + ReZero	101M	1.08

deep architectures has a large number of singular values that vanish to machine precision. While the distribution varies depending on the details of the initialization scheme, the qualitative statement holds more broadly. These results are consistent with the common observation that deep Transformer networks are extremely challenging to train.

We apply ReZero to solve the problem of poor signal propagation in Transformer layers by replacing LayerNorm and re-scaling the self-attention block. Specifically, this modifies equation (9) to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \text{sublayer}(\boldsymbol{x}_i)\,, \tag{12}$$

where α_i is the learned residual weight parameter as in the right panel of Figure 4. We share the same α_i parameter for a pair of multi-head self-attention and feed-forward network within a Transformer layer. At initialization, $\alpha_i = 0$, which allows for unimpeded signal propagation: All singular values of the input-output Jacobian are 1 and the model trivially satisfies dynamical isometry. To verify that the model remains close to dynamical isometry throughout training and for larger α_i, we show a histogram of the Jacobian singular values during the training of a 64 layer model on a toy task of language modeling on WikiText-2 [24] in Figure 5b. During training the weight of the residual connection gradually increases, allowing the Transformer to model extremely complex functions while maintaining signal propagation properties close to dynamical isometry.

5.2 Convergence speed

We pick language modeling on `enwiki8` [25] as a benchmark because strong language models are a good indicator of downstream NLP task performance [4]. Our aim in these experiments is to measure the convergence speed of each method by measuring the number of iterations it takes for a 12 layer Transformer to reach 1.2 bits per byte (BPB) on `enwiki8`.

Since the introduction of Transformers [21], there have been several competing placements of the LayerNorm within the Transformer to achieve better convergence [4, 26]. We experiment with 3 Transformer normalization methods and compare against the ReZero Transformer. The *Post-Norm* (Row 5 in Table 1) method is equivalent to the vanilla Transformer in [21], the *Pre-Norm* (Row 4 in Table 1) method was recently introduced in [26] and the *GPT2-Norm* ($\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \text{Norm}(F(\boldsymbol{x}_i))$) was used in the training of GPT2 [4], which has successfully trained Transformers up to 48 layers. Finally, we experiment with our proposed ReZero method with α initialized to either zero or one. The hyperparameters are in the appendix A.

Our results (Table 2) show that *Post-Norm* diverges during training while all other models are able to converge. This is not surprising as the original Transformer implementation required a learning rate warm-up and this is also confirmed in [26]. To verify this, we re-ran the *Post-Norm* setup with 100 steps of learning rate warm-up and find that the model is able to converge to 1.2 BPB in 13,690 iterations. Under this setting, we compared other LayerNorm placements schemes against *Post-Norm*. We find that the other placements led to initially faster convergence, but ultimately *Post-Norm* catches up in performance, resulting in relatively slower convergence for *Pre-Norm* and *GPT2-Norm*. However, other LayerNorm placements have an advantage over *Post-Norm* in that they do not require learning rate warm-up, thus have fewer hyperparameters to tune. ReZero with $\alpha = 1$

does not show an improvement over the vanilla Transformer, indicating the importance of initializing $\alpha = 0$. With our proposed initialization of $\alpha = 0$, ReZero converges 56% faster than the vanilla Transformer.

5.3 Deeper Transformers

Transformer models that achieve state of the art performance in many NLP tasks [23] usually have less than 24 layers. The deepest model as of our work used up to 78 layers [27] and requires 256 GPUs for training. In this section, we will scale beyond hundreds of Transformer layers and still remain trainable on a desktop machine. To examine whether our approach scales to deeper Transformers, we extend our 12 layer ReZero Transformer from Section 5.2 to 64 and 128 layers and compare against the vanilla Transformer (*Post-Norm*). The hyperparameters are in appendix section B.

Our results (Table 3) indicate that a 12 layer ReZero Transformer attains the same BPB as a regular Transformer after convergence, which shows that we do not lose any representational expressivity in our model by replacing LayerNorm with ReZero. We find that trying to train deep vanilla Transformers lead to either convergence difficulties or slow training times. When scaled to 64 layers, the vanilla Transformer fails to converge even with a warm-up schedule. A ReZero Transformer with initialization of $\alpha = 1$ diverges, supporting our theoretically motivated initialization at $\alpha = 0$. The deeper ReZero Transformers are able to attain better performance than the shallower Transformers.

For comparison, we also display results from Character Transformer [11] which had a similar setup for reference. However, Character Transformer uses more parameters and has many additional auxiliary losses to achieve their performance, which is orthogonal to our work. Our 128 layer Transformer achieves similar performance without any intermediate losses, uses half the number of parameters and has larger depth. We did not tune our hyperparameters, and our models can potentially achieve better results with stronger regularization and a learning rate schedule.

To probe deeper into our model, we examine the behavior of residual weights α_i during training for our 12 layer and 64 layer ReZero Transformer (Figure 6). It is useful to view $|\alpha_i|$ as the amount of contribution each layer provides to the overall signal of the network. We see that an interesting pattern emerges for both the shallow and the deeper ReZero Transformer. During the early iterations of training, the residual weights quickly increase to a peak value, then slowly decays to a small value throughout its training. Early in training, the higher layers tend to be dominant (they peak earlier) and towards the end of training each layer is utilized to a similar degree. The average $|\alpha_i|$ at the end of training is 0.0898 and 0.0185 for the 12 and 64 layer models respectively, which is approximately $1/L$, where L is the number of residual layers.

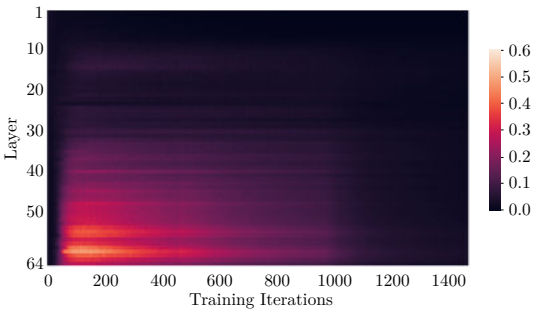

Figure 6: Heat map for residual weight $|\alpha_i|$ evolution during training for 64L ReZero Transformer.

Interestingly, this pattern also occurs in the 12 layer ReZero Transformer when we initialized $\alpha = 1$, except the model spends the first ≈ 50 iterations forcing the α's to small values, before reaching a similar pattern to that shown in Figure 6. This empirical finding supports our proposal that we should initialize $\alpha = 0$ even for shallow models.

6 Training ResNets faster

In the previous sections, we saw how ReZero connections enable training of deep networks that contain layers with vanishing Jacobian singular values, such as ReLU activations or self-attention. Some of these architectures are not trainable without ReZero connections or other architectural changes. In this section, we apply ReZero connections to deep residual networks for image recognition [2]. While these networks are trainable without ReZero connections, we observe that the validation error

for a ResNet56 model[4] trained (up to 200 epochs) on the CIFAR-10 dataset improves significantly — from $(7.37 \pm 0.06)\%$ to $(6.46 \pm 0.05)\%$ — after trading all vanilla residual connections in the model for ReZero connections[5]. The number of epochs to decrease the validation error below 15% also dropped by $(32 \pm 14)\%$ after implementing ReZero. While these results provide only limited insight by themselves, they point towards broader applicability of ReZero connections and motivate further study.

7 Conclusion

We introduced ReZero, a simple architecture modification that facilitates signal propagation in deep networks and helps the network maintain dynamical isometry. Applying ReZero to various residual architectures – fully connected networks, Transformers and ResNets – we observed significantly improved convergence speeds. Furthermore, we were able to efficiently train Transformers with hundreds of layers, which has been difficult with the original architecture. We believe deeper Transformers will open doors for future exploration.

While training models with ReZero, we discovered interesting patterns in the values of residual weights of each layer $|\alpha_i|$ over the course of training. These patterns may hint towards some form of curriculum learning and allow for progressive stacking of layers to further accelerate training [28]. Patterns of residual weights can be crucial to understand the training dynamics of such deeper networks and might be important to model performance, which we will explore in future work.

Acknowledgements

The work of TB was supported in part by DOE under grants no. DE-SC0009919 and by the Simons Foundation SFARI 560536. The work of BPM and HHM was supported by Amazon via the grant of Alexa Prize Grand Challenge 3.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[5] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[7] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[8] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

[4] For our experiments we used the implementation by Yerlan Idelbayev (available at `github.com/akamaster/pytorch_resnet_cifar10`) that very closely resembles the original architecture [2].

[5] Our setup differs from the SkipInit proposal in [20], in that we retain the BatchNorm layer.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[11] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166, 2019.

[12] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[13] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pages 4785–4795, 2017.

[14] Jeffrey Pennington, Samuel S Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. *arXiv preprint arXiv:1802.09979*, 2018.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[16] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[17] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[18] Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pages 7103–7114, 2017.

[19] Dar Gilboa, Bo Chang, Minmin Chen, Greg Yang, Samuel S Schoenholz, Ed H Chi, and Jeffrey Pennington. Dynamical isometry and a mean field theory of lstms and grus. *arXiv preprint arXiv:1901.08987*, 2019.

[20] Soham De and Samuel L Smith. Batch normalization biases deep residual networks towards shallow paths. *arXiv preprint arXiv:2002.10444*, 2020.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[22] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL*, pages 4171–4186, 2019.

[24] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

[25] Matt Mahoney. Large text compression benchmark, 2009.

[26] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*, 2020.

[27] Microsoft. Turing-nlg: A 17-billion-parameter language model, 2020.

[28] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tie-Yan Liu. Efficient training of BERT by progressively stacking. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2337–2346. PMLR, 2019.

[29] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[30] Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. Reducing BERT pre-training time from 3 days to 76 minutes. *CoRR*, abs/1904.00962, 2019.

A Convergence speed experimental hyperparameters

For all model variants in Section 5.2, we control the batch size to be 1080, number of layers to 12, feed-forward and attention dropout to 20%, hidden and embedding size to 512 units, context length to 512, the attention heads to 2, and GELU [29] activation in the point-wise feed-forward layer. To accommodate large batch training we use the LAMB optimizer [30] with a fixed learning rate of 0.016. Although learning rate schedules tend to improve performance [23], we omit them to simplify our training process.

B Deep Transformers experimental hyperparameters

In Section 5.3, in order to examine whether our approach scales to deeper Transformers, we scale our 12 layer ReZero Transformer from Section 5.2 to 64 layers and 128 layers and compare it against the vanilla Transformer (*Post-Norm*). Due to memory constraints, we decreased the hidden size from 512 to 256 and reduced batch size to 304 and 144 for the 64 layer and 128 layer model respectively. Following guidelines from [30] we also adjusted the learning rate to according to $0.0005 \times \sqrt{\text{batch size}}$. For all models in our experiments we limit training to a maximum of 100 training epochs.