ReZero is All You Need: Fast Convergence at Large Depth

Thomas Bachlechner[1] **Bodhisattwa Prasad Majumder**[2] **Henry Mao**[3] **Gary Cottrell**[2] **Julian McAuley**[2]

[1]MeetElise, USA,
[2]Computer Science and Engineering, UC San Diego, USA,
[3]Altum Inc., USA,

Abstract

Deep networks often suffer from vanishing or exploding gradients due to inefficient signal propagation, leading to long training times or convergence difficulties. Various architecture designs, sophisticated residual-style networks, and initialization schemes have been shown to improve deep signal propagation. Recently, Pennington et al. [2017] used free probability theory to show that dynamical isometry plays an integral role in efficient deep learning. We show that the simplest architecture change of gating each residual connection using a single zero-initialized parameter satisfies initial dynamical isometry and outperforms more complex approaches. Although much simpler than its predecessors, this gate enables training thousands of fully connected layers with fast convergence and better test performance for ResNets trained on an image recognition task. We apply this technique to language modeling and find that we can easily train 120-layer Transformers. When applied to 12 layer Transformers, it converges 56% faster.

1 INTRODUCTION

Deep learning has enabled significant improvements in state-of-the-art performance across domains [LeCun et al., 2015, He et al., 2016a, Klambauer et al., 2017, Radford et al., 2019]. The expressivity of neural networks typically grows exponentially with depth [Poole et al., 2016], enabling strong generalization performance, but often induces vanishing/exploding gradients and poor signal propagation through the model [He et al., 2015]. Practitioners have relied on residual [He et al., 2016a] connections along with complex gating mechanisms in highway networks [Srivastava et al., 2015], careful initialization [Mishkin and Matas, 2015, Xiao et al., 2018, Zhang et al., 2019] and normalization such as BatchNorm [Ioffe and Szegedy, 2015] and LayerNorm [Ba et al., 2016] to mitigate this issue.

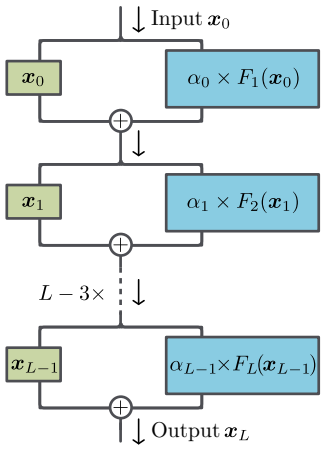

Figure 1: ReZero

Recent theoretical work [Pennington et al., 2017] applied free probability theory to randomly initialized networks and demonstrated that dynamical isometry is a key indicator of trainability. Motivated by this theory, we study the simplest modification of deep networks that ensures initial dynamical isometry, which we call ReZero. ReZero is a small addition to any network 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 a 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. ReZero provides

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1352–1361.

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

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).

several benefits:

- **Architecture agnostic:** Unlike more complex schemes, ReZero is simple and architecture agnostic, making its implementation widely applicable to any residual-style architectures without much tuning and only a few lines of code.
- **Deeper learning:** While simpler than existing approaches [Srivastava et al., 2015, Zhang et al., 2019], ReZero effectively propagates signals 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, LayerNorm [Ba et al., 2016] or auxiliary losses [Al-Rfou et al., 2019].
- **Faster convergence:** We observe 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 39% speed up to reach 80% 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 [Montufar et al., 2014, Poole et al., 2016]. 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, and residual connections. We show some of the popular ways to combine residual networks with normalization in Table 1.

2.1 CAREFUL INITIALIZATION

The dynamics of signal propagation in randomly initialized deep and wide neural networks have been formalized via mean field theory [Pennington et al., 2017, Xiao et al., 2018, Pennington et al., 2018]. 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 [Glorot and Bengio, 2010, He et al., 2015], which motivated a rescaling of the weights such that $\chi \approx 1$ and on average signal strengths are neither enhanced or attenuated.

Pennington et al. [2017, 2018] 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* [Saxe et al., 2013], 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 [Pennington et al., 2017]. For some activation functions and network architectures, elaborate initialization schemes allow the network to satisfy dynamical

isometry at initialization, which significantly improves training dynamics [Schoenholz et al., 2016, Poole et al., 2016, Yang and Schoenholz, 2017, Gilboa et al., 2019].

2.2 NORMALIZATION

An alternative approach to improve the trainability of deep networks is to incorporate layers that explicitly provide normalization. Many normalization modules have been proposed, with the two most popular being BatchNorm [Ioffe and Szegedy, 2015] and LayerNorm [Ba et al., 2016]. In general, normalization aims to ensure that initially, signals have zero mean and unit variance as they propagate through a network, reducing *covariate shift* [Ioffe and Szegedy, 2015].

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 for ResNet in [He et al., 2016a] enabled a deep residual learning framework in the context of convolutional networks for image recognition that significantly increased the trainable depth. In [He et al., 2016b] it was recognized that identity (pre-activation) residual connections allow for improved signal propagation. Residual connections in ResNets allowed for training of extremely deep networks, but the same has not been the case for Transformer architectures. Deep Transformer architectures have thus far required extreme compute budgets or auxiliary losses.

Careful initialization has been employed in conjunction with residual connections. It has been proposed to initialize residual blocks around zero in order to facilitate better signal propagation [Srivastava et al., 2015, He et al., 2016b, Goyal et al., 2017, Hardt and Ma, 2016, He et al., 2019, Zhang et al., 2019]. Another example of carefully initialized residual connections was proposed as Gated ResNets [Srivastava et al., 2015, Savarese et al., 2016]. These connections are defined by $\boldsymbol{x}_{i+1} = (1-\alpha_i)\cdot \boldsymbol{x}_i + \alpha_i \cdot F(\boldsymbol{x}_i)$, which differs from (1). Gated ResNets eliminate the skip-connection once the parameters α_i grow during training, reducing the signal propagation length of the network. These networks were an intuitively motivated proposal based on Highway networks, while ReZero is founded upon the field theoretic study of deep signal propagation. In the absence of skip connections the correlation function between input perturbations decays exponentially with depth, instead of polynomially for networks containing skip connections (such as ReZero), as explained in [Yang and Schoenholz, 2017].

Recently, SkipInit [De and Smith, 2020], 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 [De and Smith, 2020], and study more generally signal propagation in deeper networks across multiple architectures and beyond BatchNorm.

3 REZERO

We propose **ReZero**[1] (**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 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 [Vaswani et al., 2017].

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 (e.g., $L \gg 1$) network is extremely sensitive to any small perturbations of

[1] Codes for ReZero applied to various neural architectures: `https://github.com/majumderb/rezero`

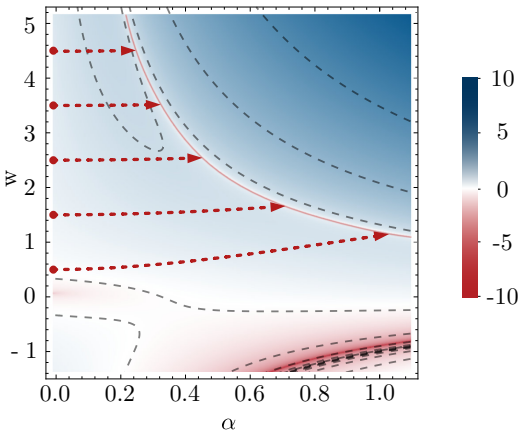

Figure 2: Contour plot of the log gradient norm, $\log\|\partial\mathcal{C}\|_2$, over the network weight w and the residual weight α during the training of the linear function $x_{L=5} = 50 \times x_0$ via gradient descent using a training set of $x_0 = \{1., 1.1, \ldots, 1.8\}$. Gradient descent trajectories initialized at $\alpha = 0$ are shown in red for five different initial w's. The trajectory dynamics avoid the poorly conditioned regions around $\alpha \approx 1$ and converge to the solution $\alpha w \approx 1.2$.

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

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

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

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

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— our experiments show that with an initial $\alpha = 0$ and a small enough learning rate, gradient descent changes the weights in a way that avoids large outputs and keeps the parameter trajectory within a feasible region while retaining the expressive power of the network. The first non-trivial steps of the residual weight updates are given by

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

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 compared to a vanilla residual network. We illustrate the training dynamics and gradients in Figure 2.

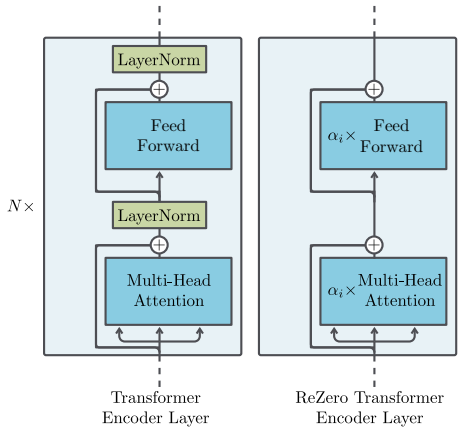

Figure 3: ReZero for Transformers

4 REZERO FOR DEEP NEURAL NETWORKS

We now introduce the application of ReZero in more complex and widely used neural network architectures such as fully-connected neural networks, ResNets and Transformers. Figure 3 shows a representative diagram when ReZero is applied to a Transformer.

Fully-connected networks A fully-connected networks with ReLU activations can be initialized with ReZero weight (α) for each unit with a skip connection as defined in row 6 of Table 1.

ResNets Residual connections enabled the first widely used feed-forward networks for image recognition with hundreds of layers [Srivastava et al., 2015, He et al., 2016a]. It was quickly realized that applying the residual connection after the activation function (in PreAct-ResNets [He et al., 2016b], see Table 1), as well as initializing the network closer to the identity mapping (in [Savarese et al., 2016, Goyal et al., 2017, Hardt and Ma, 2016, He et al., 2019, Zhang et al., 2019, De and Smith, 2020]) leads to improved performance. ReZero combines the benefits of both approaches and is the simplest implementation in this sequence of papers. Similar to fully-connected networks, every residual block in a deep convolutional residual network is added with the scalar multiplier α initialized at 0 along with the residual connection (row 6, Table 1).

Transformers Transformers architectures [Vaswani et al., 2017] recently gained significant popularity and success both in supervised and unsupervised NLP tasks [Devlin et al., 2019, Al-Rfou et al., 2019]. Transformers are built by stacking modules that first perform self-attention, then a point-wise feed-forward transformation. The original Transformer [Vaswani et al., 2017] implementation can be seen as a residual network with post-normalization (row 5 in Table

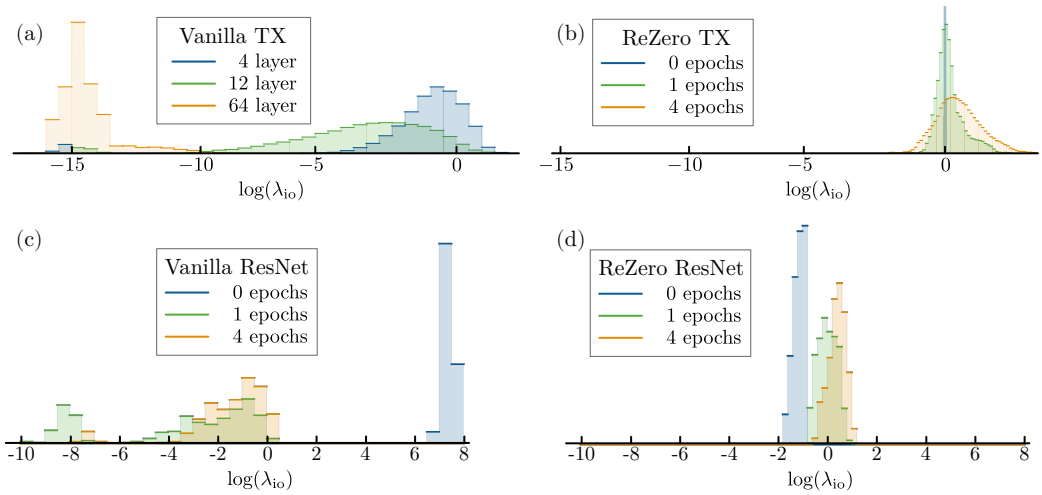

Figure 4: Histograms of log singular values ($\log(\lambda_{\text{io}})$) for the input-output Jacobian: (a) Transformer encoder at initialization; depths 4, 12 and 64 layers. No evolution during training is shown for the 64 layer Transformer because this network diverges rapidly. (b) ReZero Transformer encoder with 64 layers before/during training. Deep ReZero Transformers remain much closer to dynamical isometry, where mean singular value $\lambda_{\text{io}} \approx 1$. (c) Vanilla ResNet-110 during initial training (d) ReZero ResNet-110 during initial training.

1). Inside a Transformer module the output of each sublayer is added via a residual connection and then normalized by LayerNorm,

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

where sublayer $\in \{\text{self-attention}, \text{feed-forward}\}$, as illustrated in Figure 3. In the ReZero Transformer this relation is changed to $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \text{sublayer}(\boldsymbol{x}_i)$.

4.1 SIGNAL PROPAGATION IN REZERO-ENABLED NETWORKS

As discussed in §2.1 the perturbative signal propagation is captured by the input-output Jacobian (3), and good signal propagation is associated with dynamical isometry, where all singular values of this matrix are close to unity. We now discuss the detailed distribution of singular values for vanilla and ReZero variants of widely used deep networks: ResNets and Transformers. Qualitatively similar conclusions hold for fully connected networks.

ResNets ResNets are deep networks in which identity connections allow signal to propagate deep through a network. The signal propagation in these networks has been studied in detail [Yang and Schoenholz, 2017], and it was shown that the signal deteriorates only with the logarithm of the depth, allowing for very deep networks. However, for very deep networks the transformations of the layers still compound and lead to very large singular values of the input-output Jacobian, far from dynamical isometry. We show the distribution of singular values for a vanilla ResNets-110 in Figure 4c. This distribution shows that there exist directions in signal-space in which a perturbation gets magnified by a factor of 10^8 at initialization. After some training the bulk of the singular values moves closer to dynamical isometry at $\log(\lambda_{\text{io}}) = 0$. Figure 4d shows the singular value distribution for a ReZero ResNet-110. Both before and during training the network remains much closer to dynamical isometry. Comparing the distributions directly demonstrates that ReZero improves the signal propagation by a factor of about 10^6 compared to vanilla variants.

Transformers Two crucial components relevant to the signal propagation in the original Transformer layers include LayerNorm [Ba et al., 2016] and (multi-head) self attention [Vaswani et al., 2017]. Neither component by itself or in conjunction with a vanilla residual connection can satisfy dynamical isometry for all input signals, as we show with a theoretical argument in Appendix §A. We verify these claims in practice by evaluating the change of the attention output under an infinitesimal variation of each input element, which yields the input-output Jacobian. We show the input-output Jacobian for Transformer encoder layers of various depths with Xavier uniform initialized weights in Figure 4a. While shallow Transformers exhibit a singular value distribution peaked around unity, we clearly observe that the Jacobian of deeper Transformers 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 and is consistent with the common observation that deep Transformer networks are extremely challenging to train.

In order to facilitate deep signal propagation we apply ReZero by replacing LayerNorm and re-scaling the self-

attention block. Specifically, this modifies equation (9) to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \text{sublayer}(\boldsymbol{x}_i)\,, \tag{10}$$

where α_i is the learned residual weight parameter as in the right panel of Figure 3. 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 Transformer decoder language model on WikiText-2 [Merity et al., 2017] in Figure 4b. 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.

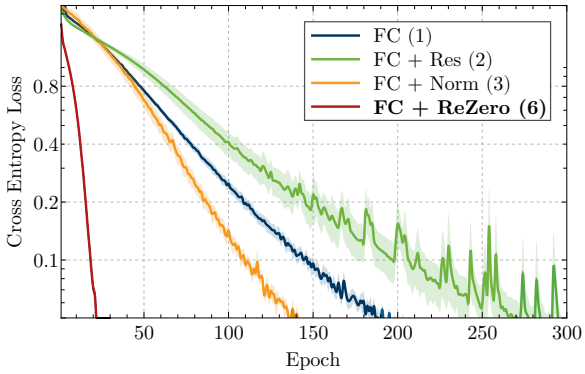

Figure 5: Cross entropy loss during training of four variants of 32 layer fully-connected networks with width 256 and ReLU activations. Numbers in parentheses refer to the architectures in the corresponding rows of Table 1. We average over five runs each and show 1σ error bands. We train using Adagrad [Duchi et al., 2011] with learning rate 0.01.

5 FASTER CONVERGENCE AT LARGE DEPTH

In this section, we will experiment with ReZero-enabled networks to investigate how these networks enables faster learning in very deep networks. We experiment with (1) fully connected ReLU networks, (2) Convolutional ResNets, and (3) Transformers.

5.1 FULLY CONNECTED NETWORKS

We study the effect of ReZero on deep ReLU networks, and compare it with some of the 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 proposed ReZero (row 6). We choose the initial weights $\boldsymbol{W}_i$ to be normally distributed with variances optimal for training [He et al., 2015, Yang and Schoenholz, 2017], e.g., $\sigma_W^2 = 2/w$ for all but the vanilla residual network where $\sigma_W^2 \approx 0.25/w$.

As a sample toy task, we train four different 32-layer network architectures on the CIFAR-10 dataset 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.

We show the evolution of the training loss in Figure 5. In our simple experiment, the ReZero architecture converges to fit the training data between 7 and 15 times faster than 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 [Yang and Schoenholz, 2017].

With increasing depth, the advantages of the ReZero architecture become more apparent. To verify that this architecture ensures trainability to large depth we successfully trained fully connected ReZero networks with up to $10,000$ layers on a laptop with one GPU[2] to overfit the training set.

5.2 CONVOLUTIONAL RESNETS

Here, we apply ReZero connections to deep convolutional residual networks for image recognition [He et al., 2016a]. While these networks are trainable without modification, we observe that ReZero accelerates training and improves accuracy.

In order to compare different methods (ResNet [He et al., 2016a] modified by Gated ResNet [Srivastava et al., 2015, Savarese et al., 2016], zero γ [Goyal et al., 2017, Hardt and Ma, 2016], FixUp [Zhang et al., 2019], ReZero and Pre-Act ResNet [He et al., 2016b] modified with ReZero) to improve deep signal propagation, we trained various versions of residual networks on the CIFAR-10 image classification dataset, each with identical hyperparameters. We discuss the architectures and hyperparameters in detail in Appendices §D and §E. In Table 2 we present results for the validation error, the number of epochs to reach 80% accuracy (to quantify the initial training speed, similar to DAWNbench [Coleman et al., 2017]), and loss on the training data. ReZero performs better than the other methods for ensuring deep signal propagation: it accelerates training as much as FixUp, but retains the regularizing properties of the BatchNorm

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

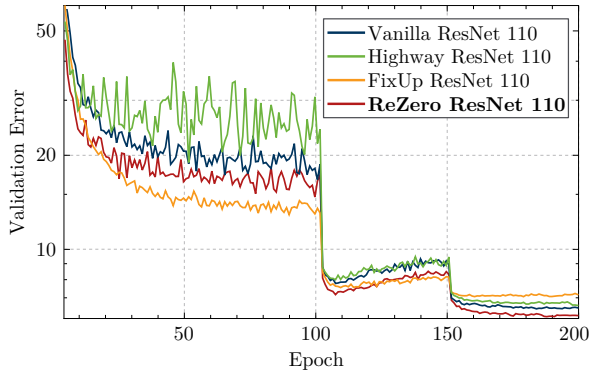

Figure 6: Validation error for four variants of ResNet-110. Details of hyperparameters are in Appendix §E.

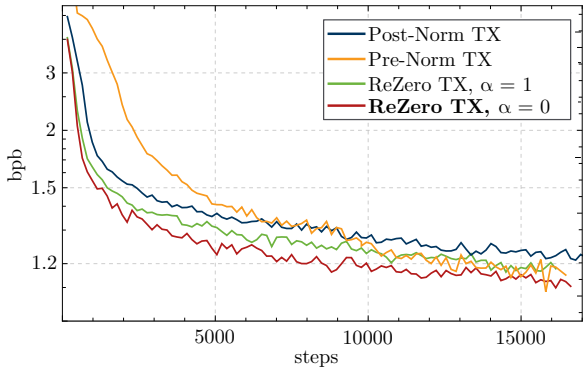

Figure 7: Bits per byte on `enwiki8` during training of three variants of 12 layer Transformers normalization variants against ReZero, as described in §5.3. Post-Norm without warm-up diverges within 100 training iterations, and is omitted from graph. Numerical comparison is in Appendix §B.

layer. Gated ResNets initially train very fast, but perform significantly worse than ReZero on test data. The complete evolution of validation error is shown in Figure 6 where we compare ReZero with vanilla ResNets, Highway networks and FixUp. We observe that at convergence, ReZero even achieves lower validation error compared to other baselines.

Model	Epochs to 80% Acc.
ResNet-110	23 ± 4
+ Gated ResNet	10 ± 2
+ zero γ	36 ± 5
+ FixUp	15 ± 1
+ **ReZero**	14 ± 1
Pre-activation ResNet-18	26 ± 2
+ **ReZero**	12 ± 1

Table 2: Comparison of ResNet variants on CIFAR-10. The uncertainties correspond to standard error. Complete table is in Appendix §E.

In order to demonstrate the accelerated training of ReZero networks, we implemented superconvergence [Smith and Topin, 2019] in a Pre-activation ResNet-18 with ReZero connections. The phenomenon of superconvergence uses one cycle of an increasing and decreasing learning rate, in which a large maximum learning rate serves as a regularizer. This yields very fast convergence for some networks. We find that the training duration to achieve 94% accuracy decreases from the 60 epochs for the baseline[3] model to 45 epochs for a ReZero model.

5.3 TRANSFORMERS

To experiment with Transformers, we select language modeling on `enwiki8` [Mahoney, 2009] as a benchmark because strong language models are a good indicator of downstream NLP task performance [Radford et al., 2019].

Convergence speed Since the introduction of Transformers [Vaswani et al., 2017], there have been several competing placements of the LayerNorm [Nguyen and Salazar, 2019, Raffel et al., 2019] within the Transformer to achieve better convergence [Radford et al., 2019, Xiong et al., 2020]. 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 [Vaswani et al., 2017], the *Pre-Norm* (Row 4 in Table 1) method was recently introduced in [Xiong et al., 2020] and the *GPT2-Norm* ($\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \text{Norm}(F(\boldsymbol{x}_i))$) was used in the training of GPT2 [Radford et al., 2019], 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. Details of our hyperparameter settings are in Appendix §B. In our preliminary experiments, we also tried with a range of α values but found that it generally isn't important so long as α is sufficiently small.

Our results show that *Post-Norm* (without warmup) 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 likely to overcome its poor initial signal propagation, as confirmed in [Xiong et al., 2020]. To verify this, we re-ran the *Post-Norm* setup

[3]Our implementation was inspired by the codes from fast.ai available at `https://github.com/fastai/imagenet-fast/tree/master/cifar10`. We replicated this model and added ReZero.

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

Table 3: Comparison of Transformers (TX) on the `enwiki8` test set. Char-TX refers to the Character Transformer [Al-Rfou et al., 2019] and uses additional auxiliary losses to achieve its performance.

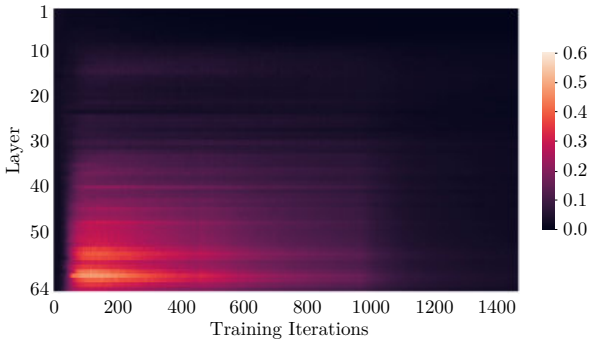

Figure 8: Heat map for residual weight $|\alpha_i|$ evolution during training for 64L ReZero Transformer.

with 100 steps of learning rate warm-up (Figure 7) and find that the model is able to converge to 1.2 BPB in 13,690 iterations. Under this setting, we compared other LayerNorm placement 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, and 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.

Deeper Transformers Deeper Transformers require significantly more compute to train, with 78 layer Transformers requiring a cluster of 256 GPUs [Microsoft, 2020]. This cost comes from an increase in memory requirements and poor signal propagation. The Character Transformer [Al-Rfou et al., 2019] mitigated this issue by having intermediate layers predict the target objective as an auxiliary loss, thus circumventing vanishing gradients. In this section, we extend our 12 layer ReZero Transformer from Section 5.3 to 64 and 128 layers and compare against the vanilla Transformer (*Post-Norm*) and the Character Transformer. 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 leads to convergence difficulties. 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.

We also display results from Character Transformer [Al-Rfou et al., 2019], which had a similar setup, but required more parameters and used intermediate and other auxiliary losses to achieve their performance. In contrast, our 128 layer Transformer achieves similar performance and learns effectively without any intermediate losses. We did not tune our hyperparameters (Appendix §C) and our models can potentially achieve better results with stronger regularization.

To probe deeper into our model, we examine the behavior of residual weights α_i during training for our 12 and 64 layer ReZero Transformers. The results for the 12 and 64 layer Transformer are qualitatively similar, and we show the 64 layer result in Figure 8. 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: During the early iterations of training, the residual weights quickly increase to a peak value, then slowly decay to a small value later in training. Early in training, the higher layers tend to be dominant (they peak earlier) and toward the end of training each layer is used 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 $1/L$. Interestingly, this pattern also occurs in the 12 layer ReZero Transformer when we initialized α to 1. The difference is that the model spends the first ≈ 50 iterations forcing the α's to small values, before reaching a similar pattern to that in Figure 8. This empirical finding supports our proposal that we should initialize $\alpha = 0$ even for shallow models.

6 CONCLUSION

We introduced ReZero, a simple architectural 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 the door to future exploration. In order to test whether zero initialization of the residual weights is important, we trained transformer networks with α initialized to either 1 or 0 and observed a significant improvement in performance for $\alpha = 0$ over both the baseline and $\alpha = 1$ initialization. This is consistent with our hypothesis that initial dynamical isometry drives performance improvement.

While training models with ReZero Transformers, 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 [Gong et al., 2019]. 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.

Author Contributions

The first three authors contributed equally, they are listed ordered by last name.

Acknowledgements

We thank anonymous reviewers for providing valuable feedback. BPM is partly supported by a Qualcomm Innovation Fellowship and NSF Award #1750063. Findings and observations are of the authors only and do not necessarily reflect the views of the funding agencies.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *AAAI*, volume 33, 2019.

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

Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102, 2017.

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

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*, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul), 2011.

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.

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

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tie-Yan Liu. Efficient training of BERT by progressively stacking. In *ICML*, Proceedings of Machine Learning Research, 2019.

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

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

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

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 2016b.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019.

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

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553), 2015.

Matt Mahoney. Large text compression benchmark, 2009. URL http://mattmahoney.net/dc/text.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

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

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

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *CoRR*, abs/1910.05895, 2019. URL http://arxiv.org/abs/1910.05895.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *NIPS*, 2017.

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

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NIPS*, pages 3360–3368, 2016.

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

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

Pedro HP Savarese, Leonardo O Mazza, and Daniel R Figueiredo. Learning identity mappings with residual gates. *arXiv preprint arXiv:1611.01260*, 2016.

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.

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

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

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

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.

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.

Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *NIPS*, 2017.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.