THE UNREASONABLE EFFECTIVENESS OF (ZERO) INITIALIZATION IN DEEP RESIDUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Normalization layers are a staple in state-of-the-art deep neural network architectures. They are widely believed to stabilize training, enable higher learning rate, accelerate convergence and improve generalization, though the reason for their effectiveness is still an active research topic. In this work, we challenge the commonly-held beliefs by showing that *none* of the perceived benefits is unique to normalization. Specifically, we propose ZeroInit, an initialization motivated by solving the exploding and vanishing gradient problem at the beginning of training by initializing as a zero function. We find training residual networks with ZeroInit to be as stable as training with normalization - even for networks with 10,000 layers. Furthermore, with proper regularization, ZeroInit *without* normalization matches or exceeds the performance of state-of-the-art residual networks in image classification and machine translation.

1 INTRODUCTION

Artificial intelligence applications have witnessed major advances in recent years (Krizhevsky et al., 2012; Hinton et al., 2012; Sutskever et al., 2014), thanks to the joint force of computing hardware development, large scale datasets as well as research and industrial community efforts. At the core of this revolution is the development of novel neural network models and their training techniques. For example, since the landmark work of He et al. (2016), most of the state-of-the-art image recognition systems are built upon a deep stack of network blocks consisting of convolutional layers and additive skip connections, with some normalization mechanism (e.g. batch normalization (Ioffe & Szegedy, 2015)) to facilitate training and generalization. Besides image classification, various normalization techniques (Ulyanov et al., 2016; Ba et al., 2016; Salimans & Kingma, 2016; Wu & He, 2018) have been found essential to achieving good performance on other tasks, such as machine translation (Vaswani et al., 2017) and generative modeling (Zhu et al., 2017). They are widely believed to have multiple benefits for training very deep neural networks, including stabilizing learning, enabling higher learning rate, accelerating convergence, and improving generalization.

Despite the enormous empirical success of training deep networks with skip connections with normalization, there is currently no general consensus on why these normalization techniques help the training process (Santurkar et al., 2018). Intrigued by this topic, in this work we study

(i) *without* normalization, can a deep residual network be trained reliably? (And if so,)
(ii) *without* normalization, can a deep residual network be trained with the same learning rate, converge at the same speed, and generalize equally well (or even better)?

Perhaps surprisingly, we find the answers to both questions are *Yes*. In particular, we show:

- **Why normalization helps training.** We derive a lower bound for the gradient norm of a residual network at initialization, which explains why with traditional initializations, normalization techniques are *essential* for training deep residual networks at maximal learning rate. (Section 2)
- **Training without normalization.** We propose ZeroInit, a method that initializes the residual branches as zero functions so they gradually fade into effect during learning. ZeroInit enables training very deep residual networks stably at maximal learning rate without normalization. (Section 3)

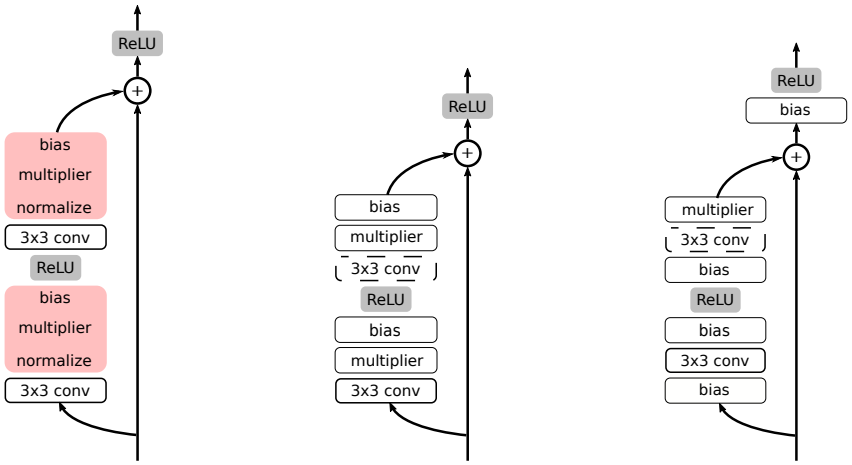

Figure 1: **Left:** ResNet basic block. Batch normalization (Ioffe & Szegedy, 2015) layers are marked in red. **Middle:** each batch normalization layer is replaced by a learned scalar multiplier followed by a scalar bias. **Right:** ZeroInit (see Section 3 for details) which re-orders the multipliers and biases. Convolution weights in boxes with dashed border are initialized as zero. Bias is disabled in all convolution layers.

- **Image classification.** We apply ZeroInit to replace batch normalization on image classification benchmarks CIFAR-10 (with Wide-ResNet) and ImageNet (with ResNet), and find ZeroInit with proper regularization matches the well-tuned baseline trained with normalization. (Section 4.3)
- **Machine translation.** We apply ZeroInit to replace layer normalization on machine translation benchmarks IWSLT and WMT using the Transformer model, and find it outperforms the baseline and achieves new state-of-the-art results. (Section 4.4)

In the remaining of this paper, we first analyze the exploding gradient problem of residual networks at initialization in Section 2. To solve this problem, we develop ZeroInit in Section 3. In Section 4 we quantify the properties of ZeroInit and compare it against state-of-the-art normalization methods on real world benchmarks. A comparison with related work is presented in Section 5.

2 Problem: ResNet with traditional initializations lead to exploding gradients

Traditional initialization methods (Glorot & Bengio, 2010; He et al., 2015; Xiao et al., 2018) attempt to set the initial parameters of the network such that the activations neither vanish nor explode. Unfortunately, it has been observed that without normalization techniques such as BatchNorm they do not account properly for the effect of residual connections and this causes exploding gradients. (Balduzzi et al., 2017) characterizes this problem for ReLU networks, and we will generalize this to residual networks with positively homogenous activation functions. A ResNet with residual blocks $\{F_1, \ldots, F_L\}$ and input $\mathbf{x}_0$ computes the activations as

$$\mathbf{x}_l = \mathbf{x}_0 + \sum_{i=0}^{l-1} F_i(\mathbf{x}_i) \tag{1}$$

Here we only consider the initialization, view the input $\mathbf{x}_0$ as fixed, and consider the randomness of the weight initialization. We will analyze the variance of each layer $\mathbf{x}_l$, denoted by $\mathrm{Var}[\mathbf{x}_l]$ (which is technically defined as the sum of the variance of all the coordinates of $\mathbf{x}_l$.) For simplicity we assume the blocks are initialized to be zero mean, i.e., $\mathbb{E}[F_l(\mathbf{x}_l) \mid \mathbf{x}_l] = 0$. By $\mathbf{x}_{l+1} = \mathbf{x}_l + F_l(\mathbf{x}_l)$, and the total law of variance, we have $\mathrm{Var}[\mathbf{x}_{l+1}] = \mathbb{E}[\mathrm{Var}[F(\mathbf{x}_l)|\mathbf{x}_l]] + \mathrm{Var}(\mathbf{x}_l)$. Resnet structure prevents $\mathbf{x}_l$ from vanishing by forcing the variance to grow with depth, i.e. $\mathrm{Var}[\mathbf{x}_l] < \mathrm{Var}[\mathbf{x}_{l+1}]$ if $\mathbb{E}[\mathrm{Var}[F(\mathbf{x}_l)|\mathbf{x}_l]] > 0$. Yet, combined with initialization methods such as He et al. (2015), the output variance of each residual branch $\mathrm{Var}[F_l(\mathbf{x}_l)|\mathbf{x}_l]$ will be about the same as its input variance

$\mathrm{Var}[\mathbf{x}_l]$, and thus $\mathrm{Var}[\mathbf{x}_{l+1}] \approx 2\mathrm{Var}[\mathbf{x}_l]$. This causes the output variance to explode exponentially with depth without normalization

$$\mathrm{Var}[\mathbf{x}_l] = \mathrm{Var}[\mathbf{x}_0] + \sum_{i=0}^{l-1} \mathrm{Var}[\mathbf{x}_i]\mathbb{E}\left[\mathrm{Var}\left[F_i\left(\frac{\mathbf{x}_i}{\sqrt{\mathrm{Var}[\mathbf{x}_i]}}\right)\middle|\, \mathbf{x}_i\right]\right] = \Omega(2^l) \tag{2}$$

for positively homogeneous blocks (see Definition 1). This is detrimental to learning because it can in turn cause gradient explosion.

As we will show, at initialization, the gradient norm of certain activations and weight tensors is *lower bounded* by the cross-entropy loss up to some constant. Intuitively, this implies that blowup in the logits will cause gradient explosion. Our result applies to convolutional and linear weights in a neural network with ReLU nonlinearity (e.g. feed-forward network, CNN), possibly with skip connections (e.g. ResNet, DenseNet), but without any normalization.

Our analysis utilizes properties of positively homogeneous functions, which we now introduce.

Definition 1 (positively homogeneous function of first degree)**.** A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is called *positively homogeneous (of first degree)* (p.h.) if for any input $\mathbf{x} \in \mathbb{R}^m$ and $\alpha > 0$, $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$.

Definition 2 (positively homogeneous set of first degree)**.** Let Θ be the set of parameters of $f(\mathbf{x})$ and $\Theta_{ph} = \{\theta_i\}_{i\in S} \subset \Theta$. We call Θ_{ph} a *positively homogeneous set (of first degree)* (p.h. set) if for any $\alpha > 0$, $f(\mathbf{x}; \Theta \setminus \Theta_{ph}, \alpha\Theta_{ph}) = \alpha f(\mathbf{x}; \Theta \setminus \Theta_{ph}, \Theta_{ph})$, where $\alpha\Theta_{ph}$ denotes $\{\alpha\theta_i\}_{i\in S}$.

Intuitively, a p.h. set is a set of parameters Θ_{ph} in function f such that for any fixed input $\mathbf{x}$ and fixed parameters $\Theta \setminus \Theta_{ph}$, $\bar{f}(\Theta_{ph}) \triangleq f(\mathbf{x}; \Theta \setminus \Theta_{ph}, \Theta_{ph})$ is a p.h. function.

Examples of p.h. functions are ubiquitous in neural networks, including various kinds of linear operations without bias (fully-connected (FC) and convolution layers, pooling, addition, concatenation and dropout etc.) as well as ReLU nonlinearity. Moreover, we have the following claim:

Proposition 1. *A function that is the composition of p.h. functions is itself positively homogeneous.*

We study classification problems with c classes and the cross-entropy loss. We use f to denote a neural network function except for the softmax layer. Cross-entropy loss is defined as $\ell(\mathbf{z}, \mathbf{y}) \triangleq -\mathbf{y}^T(\mathbf{z} - \texttt{logsumexp}(\mathbf{z}))$ where $\mathbf{y}$ is the one-hot label vector, $\mathbf{z} \triangleq f(\mathbf{x}) \in \mathbb{R}^c$ is the logits where z_i denotes its i-th element, and $\texttt{logsumexp}(\mathbf{z}) \triangleq \log\left(\sum_{i\in[c]} \exp(z_i)\right)$. Consider a minibatch of training examples $\mathcal{D}_M = \{(\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}_{m=1}^M$ and the average cross-entropy loss $\ell_{\text{avg}}(\mathcal{D}_M) \triangleq \frac{1}{M}\sum_{m=1}^M \ell(f(\mathbf{x}^{(m)}), \mathbf{y}^{(m)})$, where we use $^{(m)}$ to index quantities referring to the m-th example. $\|\cdot\|$ denotes any valid norm. We only make the following assumptions about the network f:

1. f is a sequential composition of network blocks $\{f_i\}_{i=1}^L$, i.e. $f(\mathbf{x}_0) = f_L(f_{L-1}(\ldots f_1(\mathbf{x}_0)))$, each of which is composed of p.h. functions.
2. Weight elements in the FC layer are i.i.d. sampled from a zero-mean symmetric distribution.

These assumptions hold at initialization if we remove all the normalization layers in a residual network with ReLU nonlinearity, assuming all the biases are initialized at 0.

Our results are summarized in the following two theorems, whose proofs are listed in the appendix:

Theorem 1. *Denote the input to the i-th block by $\mathbf{x}_{i-1}$. With Assumption 1, we have*

$$\left\|\frac{\partial \ell}{\partial \mathbf{x}_{i-1}}\right\| \geq \frac{\ell(\mathbf{z}, \mathbf{y}) - H(\mathbf{p})}{\|\mathbf{x}_{i-1}\|}, \tag{3}$$

where $\mathbf{p}$ is the softmax probabilities and H denotes the Shannon entropy.

Since $H(\mathbf{p})$ is upper bounded by $\log(c)$ and $\|\mathbf{x}_{i-1}\|$ is small in the lower blocks, blowup in the loss will cause large gradient norm with respect to the lower block input. Our second theorem proves a lower bound on the gradient norm of a p.h. set in a network.

Theorem 2. *With Assumption 1, we have*

$$\left\|\frac{\partial \ell_{\text{avg}}}{\partial \Theta_{ph}}\right\| \geq \frac{1}{M\|\Theta_{ph}\|}\sum_{m=1}^M \ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \triangleq G(\Theta_{ph}). \tag{4}$$

Furthermore, with Assumptions 1 and 2, we have

$$\mathbb{E}G(\Theta_{ph}) \geq \frac{\mathbb{E}[\max_{i\in[c]} z_i] - \log(c)}{\|\Theta_{ph}\|}. \tag{5}$$

It remains to identify such p.h. sets in a neural network. Here we provide three examples of p.h. sets in a ResNet without normalization: (1) the first convolution layer before max pooling; (2) the union of a spatial downsampling convolution layer in a skip connection and a convolution layer in its corresponding residual branch; (3) the fully connected layer before softmax. Theorem 2 suggests that these layers would suffer from the exploding gradient problem, if the logits $\mathbf{z}$ blow up at initialization, which unfortunately would occur in a ResNet without normalization if initialized in a traditional way. This motivates us to introduce a new initialization in the next section.

3 ZeroInit: How to effectively initialize as a zero function

Scale of the output. To solve the exploding gradient problem we have just outlined, we must ensure the output does not blow up at initialization. This provides us with the first design principle of a good initialization

(a.) The scale of the output should be independent of depth, i.e., $\mathbb{E}[\max_{i\in[c]} z_i] = \mathcal{O}(1)$

A naive approach to achieve this is to initialize the output layer and the last layer in each residual branch to 0. The network now represents the 0 function, which is indeed independent of depth at initialization. However, as we start training each of the layers will then be updated with a gradient of norm $\mathcal{O}(1)$ at the first few iterations, and will contribute a multiplicative factor of $(1 + \mathcal{O}(1))$ to the scale of output in the next forward pass, which will still lead to explosion if the depth L is large. This is again caused by the output scale of the blocks growing exponentially as depth increases.

Scale of the residual branches. We must prevent the scale of the residual branches from exploding with depth in the first training steps. This gives us our second design principle

(b.) The scale of the residual branches should be balanced, i.e., $\mathrm{Var}[F_l(\mathbf{x}_l)] = \mathcal{O}(\frac{1}{L})$

Given a network f with L residual blocks, we can thus work out a reasonably good initialization for a residual branch with m layers, by applying the above principles. In particular, we wish to initialize one layer as 0 and scale the other $m-1$ layers by λ, so that at the early training stage, the output of this branch scales its input by $\mathcal{O}(\frac{1}{L})$. Assuming the error signal passing to the branch is $\mathcal{O}(1)$, then the update to the zero-initialized layer is $\mathcal{O}(\lambda^{m-1})$, and the overall scaling of the residual branch after update is $\mathcal{O}(\lambda^{2m-2})$. We therefore set λ to $\sqrt[2-2m]{L}$ to get the desired $\mathcal{O}(\frac{1}{L})$ scaling.

Put together, we propose the following method to train residual networks without normalization:

ZeroInit (or: How to train a deep residual network without normalization)

1. Initialize the classification layer and the last convolution of each residual branch to 0.
2. Initialize every other layer traditionally, e.g. He et al. (2015), and scale only the convolutions inside residual branches by $\sqrt[2-2m]{L}$.
3. Add a scalar multiplier (initialized at 1) in every branch and a scalar bias (initialized at 0) before each convolution, linear, and element-wise activation layer.

Using ZeroInit in the ResNet of Equation 1 prevents the variance at initialization from exploding and in fact it is independent of depth, even after a few updates; it also prevents gradient vanishing in the residual branches. In constrast, batch normalization and other normalization techniques only manage to reduce this growth to a linear scaling $\mathrm{Var}[\mathbf{x}_l] = O(l)$, as observed by (Balduzzi et al., 2017). Surprisingly, we will show that properly addressing the initialization problem successfully enables the training of very deep networks without any normalization. Though we recommend increasing the strength of regularization to improve generalization. We note that ZeroInit could also

Dataset	Method	10	64	Ratio
CIFAR-10	Baseline	84.29 ± 20.82	$8\text{e}{+7} \pm 2\text{e}{+7}$	$9\text{e}{+5} \pm 4\text{e}{+5}$
	$\sqrt{1/2}$-scaling	2.24 ± 0.35	0.47 ± 0.09	0.20 ± 0.07
	LSUV	7.21 ± 1.05	31.23 ± 1.48	4.33 ± 0.83
	BatchNorm	0.74 ± 0.04	0.99 ± 0.07	1.3 ± 0.16
	ZeroInit	1.86 ± 0.30	2.30 ± 0.47	1.23 ± 0.45
SVHN	Baseline	77.92 ± 31.88	$9\text{e}{+7} \pm 6\text{e}{+7}$	$1\text{e}{+6} \pm 1\text{e}{+6}$
	$\sqrt{1/2}$-scaling	7.78 ± 4.34	2.69 ± 1.02	0.34 ± 0.32
	LSUV	12.28 ± 0.40	55.67 ± 1.85	4.53 ± 0.29
	BatchNorm	0.54 ± 0.02	1.22 ± 0.15	2.25 ± 0.36
	ZeroInit	3.46 ± 0.68	3.30 ± 0.59	0.95 ± 0.35

Table 1: Norm of the Hessian $\|\mathbf{H}\|_2$ across various initialization across two depths. Like BatchNorm, ZeroInit keep the norm relatively constant as we increase depth, which suggests that comparatively high learning rates can be used without causing divergence.

apply to other networks with skip connections like DenseNet[1] (Huang et al., 2017), but we leave this to future work.

4 EXPERIMENTS

4.1 QUANTIFYING THE LOSS SURFACES AT INITIALIZATION

We propose to contrast the structure of the loss surface at different initialization points by monitoring the norm of the Hessian. This is an approximation of the maximum learning rate (LeCun et al., 2012) and so we argue a good initialization should keep that statistic constant so the same learning rate can be used regardless of depth. It can be obtained efficiently using the power method and the R-operator (Pearlmutter, 1994) to perform fast Hessian-vector products. In these experiments, we set the tolerance of the power method to $1\text{e}{-5}$ and we use WideResnet models (Zagoruyko & Komodakis, 2016) with width 1 and we compute the Hessian over a subset of the first 1024 examples from the datasets. We compute the batch statistics required by BatchNorm using all 1024 instead of using mini-batches so the BatchNorm results represent the best-case scenario. We average our results over 3 different initialization trials for each method.

Table 1 shows that the norm of the Hessian as we increase the depth of the model from 10 to 64 layers. The baseline model, which has no normalization and uses traditional initialization (He et al., 2015), has a very large ratio between the norms at depth 10 and 64 as predicted by our analysis in Section 2. By contrast we can see that the ratio is small for BatchNorm and ZeroInit, which indicates the sharpness of the loss surface is not very affected by the increasing depth. While $\sqrt{1/2}$-scaling and LSUV Mishkin & Matas (2015) do prevent exponential growth of the norm, they are both significantly affected by depth. These results are consistent with our observations in Section 5.

4.2 TRAINING AT INCREASING DEPTH

One of the key advatanges of BatchNorm is that it leads to fast training even for very deep models (Ioffe & Szegedy, 2015). Here we will determine if we can match this desirable property by relying only on proper initialization. We propose to evaluate how each method affects training very deep nets by measuring the accuracy at the first epoch as we increase depth. We will use WideResnet models with width 1 and the default weight decay $5\text{e}{-4}$. We specifically use the default learning rate of 0.1 because the ability to use high learning rates is considered to be important to the success of BatchNorm. We use the default batch size of 128 up to 1000 layers, and use a batch size of 64 for 10,000 layers. We limit our budget of epochs to 1 due to the computational strain of evaluating models with up to 10,000 layers.

[1]In fact, thanks to replacing addition with average pooling, DenseNet does not suffer from the exploding gradient problem when training without normalization. Our preliminary experiments confirm this observation.

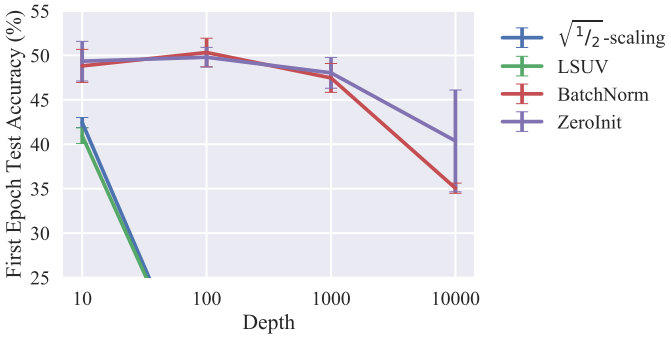

Figure 2: Depth of residual networks versus test accuracy at the first epoch for various methods on CIFAR-10 with the default BatchNorm learning rate. We observe that ZeroInit is able to train very deep networks with the same learning rate as batch normalization.

Dataset	Model	Normalization	Test Error (%)
CIFAR-10	ResNet-110 w/ BatchNorm (He et al., 2016)	Yes	6.61*
	(Zagoruyko & Komodakis, 2016)		3.8
	(Yamada et al., 2018)		**2.31**
	BatchNorm + Mixup + Cutout		2.45 ± 0.05
	ResNet-110 w/ Xavier Init (Shang et al., 2017)	No	7.78*
	ResNet-110 w/ ZeroInit		7.24*
	(Graham, 2014)		3.47
	ZeroInit + Mixup + Cutout		$\mathbf{2.34 \pm 0.01}$
SVHN	(Zagoruyko & Komodakis, 2016)	Yes	1.54
	(DeVries & Taylor, 2017)		**1.3**
	BatchNorm + Mixup + Cutout		1.38
	(Lee et al., 2016)	No	1.69
	ZeroInit + Mixup + Cutout		**1.41**

Table 2: Results on CIFAR-10, SVHN datasets. Results with * are mean/median of 5 runs.

Figure 2 shows the test accuracy at the first epoch as depth increases. Observe that ZeroInit matches or exceeds the performance of BatchNorm at the first epoch, even with 10,000 layers. LSUV and $\sqrt{1/2}$-scaling are not able to train with the same learning rate as BatchNorm past 100 layers.

4.3 Image classification

In this section, we evaluate the ability of ZeroInit to replace batch normalization in image classification applications. On the CIFAR-10 dataset, we perform experiments with ResNet-110 (He et al., 2016) and WideResnets 40-10 and on SVHN we use WideResnet 16-12 (Zagoruyko & Komodakis, 2016) with the default hyper-parameters. Except for the experiments with ResNet-110, we use Mixup (Zhang et al., 2017) and Cutout (DeVries & Taylor, 2017) with default hyper-parameters as additional regularization. On ImageNet, we base our experiments on ResNet-50 (He et al., 2016) with default hyper-parameters. We find it was beneficial to reduce the learning rate of the scalar multiplier and bias by 10x when large regularization is used on Imagenet. We found through cross-validation that ZeroInit benefits from a larger Mixup coefficient of 0.7, while the optimal coefficient for GroupNorm (Wu & He, 2018) is 0.1.

We observe in Table 2 that models trained with ZeroInit and strong regularization are competitive with state-of-the-art methods on CIFAR-10 and SVHN, as well as our baseline with batch normalization. In Table 3 and Figures 3 and 4 we show results of different methods on the ImageNet dataset with the benchmark ResNet-50 architecture. Figure 3 shows that without additional regularization ZeroInit fits the training set very well, but overfits significantly. We see in Figure 4 that ZeroInit is

Method	Normalization	Test Error (%)
BatchNorm + Mixup	Yes	**23.69**
GroupNorm + Mixup		23.86
Xavier Init (Shang et al., 2017)	No	31.48
ZeroInit		27.60
ZeroInit + Mixup		**23.96**

Table 3: ImageNet test results using the ResNet-50 architecture.

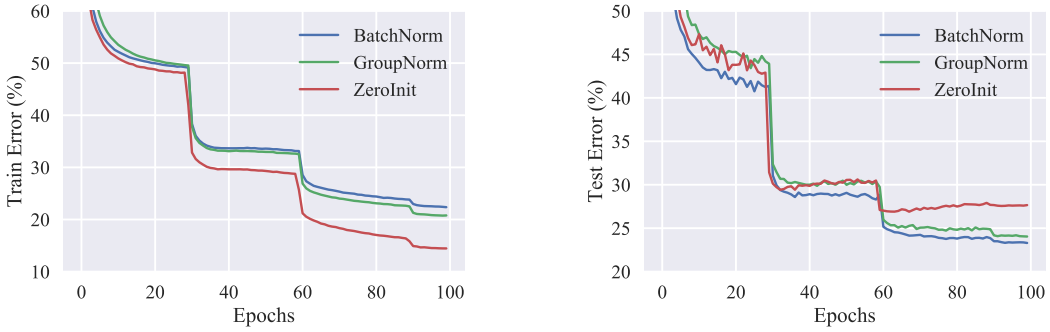

Figure 3: Training and test errors on Imagenet for various methods without additional regularization. We observe that ZeroInit is able to better fit the training data and that leads to overfitting - more regularization is needed. Results of BatchNorm and GroupNorm reproduced from (Wu & He, 2018).

competitive with networks trained with regularization when the Mixup regularizer is used. We also note that on both CIFAR-10 with ResNet-110 and ImageNet with ResNet-50, simply using ZeroInit significantly improves previous results using Xavier initialization (Shang et al., 2017).

4.4 Machine translation

To demonstrate the generality of ZeroInit, we also apply it to replace layer normalization (Ba et al., 2016) in Transformer (Vaswani et al., 2017), a state-of-the-art neural network for machine translation. Specifically, we use the fairseq library (Gehring et al., 2017b) and we follow our recommended template in Section 3 to modify the baseline model. We evaluate on two standard machine translation datasets, IWSLT German-English (de-en) and WMT English-German (en-de) following the setup of Ott et al. (2018). For the IWSLT de-en dataset, we cross-validate the dropout probability from $\{0.3, 0.4, 0.5, 0.6\}$ and find 0.5 to be optimal for both ZeroInit and the LayerNorm baseline. For the WMT'16 en-de dataset, we use the default dropout probability 0.3 for both models.

It was observed (Chen et al., 2018) that "Layer normalization is most critical to stabilize the training process... removing layer normalization results in unstable training runs". However we find training with ZeroInit to be very stable and as fast as the baseline model. Results are shown in Table 4. On the IWSLT dataset, our baseline model already outperforms several recently published state-of-the-art results, but more surprisingly, ZeroInit further improves the baseline BLEU score by more than 0.2 without any normalization mechanism, achieving a new state-of-the-art of 34.4. On the WMT dataset, ZeroInit is also competitive among the best reported results for the Transformer model.

5 Related Work

Understanding how to train very deep neural networks has received a lot of attention in the past years. Some recent work studies the effects of initialization on the learning dynamics of deep linear networks (Saxe et al., 2013), feed-forward networks (Hanin, 2018), convolutional networks (Xiao et al., 2018) or residual networks (Balduzzi et al., 2017; Hanin & Rolnick, 2018). Other work attempts to understand the effects of (batch) normalization on optimization (Hoffer et al., 2018; Santurkar et al., 2018; Bjorck et al., 2018; van Laarhoven, 2017). In contrast, our analysis in Section 2 is more gen-

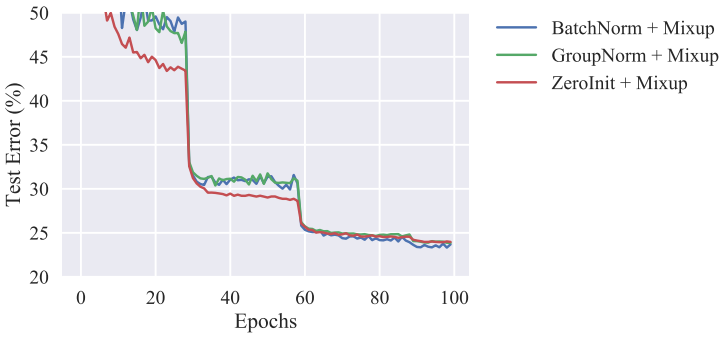

Figure 4: Test error of various methods on ImageNet with Mixup (Zhang et al., 2017). ZeroInit closely matches the final results yielded by the use of GroupNorm, without any normalization.

Dataset	Model	Normalization	BLEU
IWSLT DE-EN	(Deng et al., 2018)	Yes	33.1
	LayerNorm		34.2
	ZeroInit	No	**34.4**
WMT EN-DE	(Vaswani et al., 2017)	Yes	28.4
	LayerNorm (Ott et al., 2018)		**29.3**
	ZeroInit	No	29.0

Table 4: Comparison of ZeroInit vs. Layer Normalization for machine translation tasks.

eral and also requires weaker assumptions; it also motivates our improved initialization for residual networks in Section 3 that achieves comparable performance as (batch) normalization.

Gehring et al. (2017a); Balduzzi et al. (2017) concurrently proposed to address the initialization problem of residual nets by using the recurrence $\mathbf{x}_l = \sqrt{1/2}(\mathbf{x}_{l-1} + F_l(\mathbf{x}_{l-1}))$. This gives us a constant variance $\mathrm{Var}[\mathbf{x}_l] = \mathcal{O}(1)$, but unrolling the recurrence $\mathbf{x}_l = 0.5^{\frac{l}{2}}\mathbf{x}_0 + \sum 0.5^{\frac{l-i+1}{2}} F_l(\mathbf{x}_l)$ reveals the residual connections now vanish exponentially with depth, which defeats their purpose. Mishkin & Matas (2015) attempts to replicate the BatchNorm initialization by dividing the weight parameters of the layers by the observed standard deviation of their output using the first mini-batch. This achieves the linear scaling of the variance obtained by BatchNorm, but leads the variance of the parameters at different depth $\mathrm{Var}[\mathbf{W}_l] = O(1/l)$ to have very different scales for deep nets - which can hurt optimization. Goyal et al. (2017) proposed to initialize only the residual blocks to zero in combination with batch normalization. In line with their theory, Hardt & Ma (2016) proposed to initialize the residual branch with small Gaussian noise. While these two works attempt to (approximately) zero the residual branches, they do not address the learning dynamics issues required to allow training very deep networks without pre-training and without normalization.

6 Conclusion

In this work, we study how to train a deep residual network reliably without normalization. Our theory in Section 2 suggests that the exploding gradient problem at initialization in a positively homogeneous network such as ResNet is directly linked to the blowup of logits. Inspired by our theory, we propose ZeroInit, which initializes a deep residual network as a shallow network and allows the residual branches to gradually fade into effect during learning. Extensive experiments on real world datasets demonstrate that ZeroInit matches or outperforms normalization techniques in training deep residual networks, and achieves state-of-the-art test performance with proper regularization.

Our work opens up new possibilities for both theory and applications. On the theory side, removing the normalization layers is supposed to simplify the analysis of these residual networks. Our empirical results suggest that some previous hypotheses (e.g. Ioffe & Szegedy, 2015; Santurkar et al., 2018) about the effects of (batch) normalization may need to be revised. It would also be interesting

to understand the regularization benefits of various normalization methods. On the application side, it may be possible to develop better regularization methods, which, when combined with ZeroInit, yield improvements over the state-of-the-art.

REFERENCES

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

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*, 2017.

Johan Bjorck, Carla Gomes, and Bart Selman. Understanding batch normalization. *arXiv preprint arXiv:1806.02375*, 2018.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M Rush. Latent alignment and variational attention. *Thirty-second Conference on Neural Information Processing Systems (NIPS)*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017a.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*, 2017b.

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*, pp. 249–256, 2010.

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.

Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *arXiv preprint arXiv:1803.01719*, 2018.

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 *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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*, pp. 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. *arXiv preprint arXiv:1803.01814*, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pp. 464–472, 2016.

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

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.

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.

Wenling Shang, Justin Chiu, and Kihyuk Sohn. Exploring normalization in deep residual networks with concatenated rectified linear units. In *AAAI*, pp. 1509–1516, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

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*, pp. 5998–6008, 2017.

Yuxin Wu and Kaiming He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

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.

Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *arXiv preprint arXiv:1802.02375*, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

A PROOFS

A.1 GRADIENT NORM LOWER BOUND FOR THE INPUT TO A NETWORK BLOCK

Proof of Theorem 1. We use $f_{i\to j}$ to denote the composition $f_j \circ f_{j-1} \circ \cdots \circ f_i$, so that $\mathbf{z} = f_{i\to L}(\mathbf{x}_{i-1})$ for all $i \in [L]$. Note that $\mathbf{z}$ is p.h. with respect to the input of each network block, i.e. $f_{i\to L}((1+\epsilon)\mathbf{x}_{i-1}) = (1+\epsilon)f_{i\to L}(\mathbf{x}_{i-1})$ for $\epsilon > -1$. This allows us to compute the gradient of the cross-entropy loss with respect to the scaling factor ϵ at $\epsilon = 0$ as

$$\frac{\partial}{\partial \epsilon}\ell(f_{i\to L}((1+\epsilon)\mathbf{x}_{i-1}), \mathbf{y})\bigg|_{\epsilon=0} = \frac{\partial \ell}{\partial \mathbf{z}}\frac{\partial f_{i\to L}}{\partial \epsilon} = -\mathbf{y}^T\mathbf{z} + \mathbf{p}^T\mathbf{z} = \ell(\mathbf{z},\mathbf{y}) - H(\mathbf{p}) \tag{6}$$

Since the gradient L_2 norm $\|\partial\ell/\partial\mathbf{x}_{i-1}\|$ must be greater than the directional derivative $\frac{\partial}{\partial t}\ell(f_{i\to L}(\mathbf{x}_{i-1} + t\frac{\mathbf{x}_{i-1}}{\|\mathbf{x}_{i-1}\|}), \mathbf{y})$, defining $\epsilon = t/\|\mathbf{x}_{i-1}\|$ we have

$$\left\|\frac{\partial \ell}{\partial \mathbf{x}_{i-1}}\right\| \geq \frac{\partial}{\partial \epsilon}\ell(f_{i\to L}(\mathbf{x}_{i-1} + \epsilon\mathbf{x}_{i-1}), \mathbf{y})\frac{\partial \epsilon}{\partial t} = \frac{\ell(\mathbf{z},\mathbf{y}) - H(\mathbf{p})}{\|\mathbf{x}_{i-1}\|}. \tag{7}$$

□

A.2 GRADIENT NORM LOWER BOUND FOR POSITIVELY HOMOGENEOUS SETS

Proof of Theorem 2. The proof idea is similar. Recall that if Θ_{ph} is a p.h. set, then $\bar{f}^{(m)}(\Theta_{ph}) \triangleq f(\mathbf{x}^{(m)}; \Theta \setminus \Theta_{ph}, \Theta_{ph})$ is a p.h. function. We therefore have

$$\frac{\partial}{\partial \epsilon}\ell_{\text{avg}}(\mathcal{D}_M; (1+\epsilon)\Theta_{ph})\bigg|_{\epsilon=0} = \frac{1}{M}\sum_{m=1}^{M}\frac{\partial \ell}{\partial \mathbf{z}^{(m)}}\frac{\partial \bar{f}^{(m)}}{\partial \epsilon} = \frac{1}{M}\sum_{m=1}^{M}\ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \tag{8}$$

hence we again invoke the directional derivative argument to show

$$\left\|\frac{\partial \ell_{\text{avg}}}{\partial \Theta_{ph}}\right\| \geq \frac{1}{M\|\Theta_{ph}\|}\sum_{m=1}^{M}\ell(\mathbf{z}^{(m)}, \mathbf{y}^{(m)}) - H(\mathbf{p}^{(m)}) \triangleq G(\Theta_{ph}). \tag{9}$$

In order to estimate the scale of this lower bound, recall the FC layer weights are i.i.d. sampled from a symmetric, mean-zero distribution, therefore $\mathbf{z}$ has a symmetric probability density function with mean $\mathbf{0}$. We hence have

$$\mathbb{E}\ell(\mathbf{z},\mathbf{y}) = \mathbb{E}[-\mathbf{y}^T(\mathbf{z} - \texttt{logsumexp}(\mathbf{z}))] \geq \mathbb{E}[\mathbf{y}^T(\max_{i\in[c]} z_i - \mathbf{z})] = \mathbb{E}[\max_{i\in[c]} z_i] \tag{10}$$

where the inequality uses the fact that $\texttt{logsumexp}(\mathbf{z}) \geq \max_{i\in[c]} z_i$; the last equality is due to $\mathbf{y}$ and $\mathbf{z}$ being independent at initialization and $\mathbb{E}\mathbf{z} = \mathbf{0}$. Using the trivial bound $\mathbb{E}H(\mathbf{p}) \leq \log(c)$, we get

$$\mathbb{E}G(\Theta_{ph}) \geq \frac{\mathbb{E}[\max_{i\in[c]} z_i] - \log(c)}{\|\Theta_{ph}\|} \tag{11}$$

which shows that the gradient norm of a p.h. set is of the order $\Omega(\mathbb{E}[\max_{i\in[c]} z_i])$ at initialization. □