Batch Normalization Biases Deep Residual Networks Towards Shallow Paths

Soham De [1] Samuel L. Smith [1]

arXiv:2002.10444v1 [cs.LG] 24 Feb 2020

Abstract

Batch normalization has multiple benefits. It improves the conditioning of the loss landscape, and is a surprisingly effective regularizer. However, the most important benefit of batch normalization arises in residual networks, where it dramatically increases the largest trainable depth. We identify the origin of this benefit: At initialization, batch normalization downscales the residual branch relative to the skip connection, by a normalizing factor proportional to the square root of the network depth. This ensures that, early in training, the function computed by deep normalized residual networks is dominated by shallow paths with well-behaved gradients. We use this insight to develop a simple initialization scheme which can train very deep residual networks without normalization. We also clarify that, although batch normalization does enable stable training with larger learning rates, this benefit is only useful when one wishes to parallelize training over large batch sizes. Our results help isolate the distinct benefits of batch normalization in different architectures.

1. Introduction

The combination of skip connections (He et al., 2016a;b) and batch normalization (Ioffe & Szegedy, 2015) dramatically increases the largest trainable depth of neural networks. This has led to a rapid improvement in model performance in recent years (Tan & Le, 2019; Xie et al., 2019). While some authors have succeeded in training very deep networks without normalization layers or skip connections (Saxe et al., 2013; Xiao et al., 2018), these papers required orthogonal initialization schemes which are not compatible with ReLU activation functions. In contrast, batch normalized residual networks have been trained with thousands of layers without requiring careful initialization tricks (He et al., 2016a;b). A number of other normalization variants have also been proposed (Ulyanov et al., 2016; Salimans & Kingma, 2016; Ba et al., 2016; Wu & He, 2018; Singh & Krishnan, 2019). Following the introduction of layer normalization (Ba et al., 2016) and the growing popularity of transformers (Vaswani et al., 2017; Radford et al., 2019), almost all state-of-the-art networks currently contain both skip connections and normalization layers. However to our knowledge, despite their popularity, there is still not a simple explanation for why very deep normalized residual networks are trainable.

Our contributions. The main contribution of this paper is to provide a simple explanation of why normalization layers enable us to train deep residual networks. By viewing a residual network as an ensemble of paths with shared weights but different depths (similar to Veit et al. (2016)), we show how batch normalization ensures that, early in training, very deep residual networks with tens of thousands of layers are dominated by shallow paths containing only tens of layers. This occurs because batch normalization downscales the residual branch relative to the skip connection, by a factor proportional to the square root of the network depth. This provides an intuitive account of why deep normalized residual networks can be efficiently optimized early in training, since they behave like ensembles of shallow networks with well behaved gradients. Our analysis is related to recent work studying the initialization conditions that make deep residual networks trainable (Hanin & Rolnick, 2018; Zhang et al., 2019), but these papers did not identify how normalization layers ameliorate bad initialization choices.

The observation above suggests that one should be able to train deep residual networks without normalization or careful initialization, simply by downscaling the residual branch. To verify this claim, we introduce a one-line code change that can train very deep residual networks without normalization ("SkipInit"). Combined with additional regularization, SkipInit networks are competitive with their batch normalized counterparts at typical batch sizes (e.g. batch sizes below 1024 for ResNet-50 on ImageNet). SkipInit is similar in some aspects to the recently proposed Fixup initialization (Zhang et al., 2019). However, Fixup contains a number of components, and the relationship between these components and the effects of batch normalization layers is not clear. Our primary intention in introducing SkipInit is to provide additional evidence to support our explanation of why deep normalized residual networks are trainable.

[1]DeepMind, London. Correspondence to: Soham De <sohamde@google.com>, Samuel Smith <slsmith@google.com>.

Finally, we provide a detailed empirical analysis to help isolate the different benefits of batch normalization for both shallow and deep residual networks on CIFAR-10 and ImageNet. Our results demonstrate that, as well as enabling us to train very deep residual networks, batch normalization also increases the largest stable learning rate of shallow networks. However, contrary to previous work claiming that enabling large learning rates is the primary benefit of batch normalization (Santurkar et al., 2018; Bjorck et al., 2018), we show that this effect does not explain why batch normalized shallow networks achieve higher test accuracies than unnormalized networks under constant epoch budgets. Large learning rates are only beneficial during large batch training (Shallue et al., 2018; McCandlish et al., 2018), while for smaller batch sizes, batch normalized networks and unnormalized networks have similar optimal learning rates. These results further demonstrate the importance of evaluating the performance of alternatives to batch normalization at a wide range of batch sizes. We also verify that batch normalization can have an additional regularization effect, and we show that this benefit can be tuned by optimizing the "ghost batch size" (Hoffer et al., 2017) (the number of examples over which one computes batch statistics).

Layout of the paper. We discuss background material and related work in section 2, before introducing our analysis of deep normalized residual networks in section 3. This analysis demonstrates why networks containing identity skip connections and batch normalization layers are dominated by shallow paths early in training. We introduce SkipInit in section 4, a simple initialization scheme that can train deep residual networks without normalization. We also discuss the similarities between SkipInit and Fixup. To study the different benefits of batch normalization, in section 5 we provide an empirical evaluation of Wide-ResNets on CIFAR-10 with and without batch normalization at a wide range of batch sizes, learning rates, and network depths. We provide an empirical comparison of batch normalization, SkipInit and Fixup on ImageNet in section 6.

2. Background and related work

Residual Networks (ResNets). ResNets contain a sequence of residual blocks, which are composed of a "residual branch" comprising a number of convolutions, normalization layers and non-linearities, as well as a "skip connection", which is typically just the identity. These skip connections create pathways through the network which have a shorter effective depth than the path through all the residual branches. See figure 1 for an illustration of a residual block. Most of our experiments will follow the popular Wide-ResNet architecture (Zagoruyko & Komodakis, 2016). We also consider ResNet50-V2 (He et al., 2016b) in section 6 and ResNet50-V1 (He et al., 2016a) in appendix E.

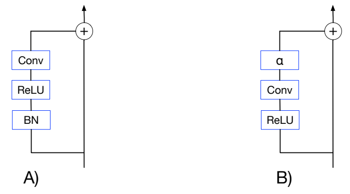


Figure 1. A) A residual block with batch normalization. It is common practice to include two convolutions on the residual branch; we show one convolution for simplicity. B) SkipInit replaces batch normalization by a single learnable scalar α. We initialize $\alpha \lesssim 1/\sqrt{d}$, where d denotes the total number of residual blocks.

Batch Normalization. As in most previous work, we apply batch normalization to convolutional layers. The inputs to and outputs from batch normalization layers are therefore 4-dimensional tensors, which we denote by $I_{b,x,y,c}$ and $O_{b,x,y,c}$. Here b denotes the minibatch, c denotes the channels, and x and y denote the two spatial dimensions. Batch normalization applies the same normalization to every input in the same channel (Ioffe & Szegedy, 2015):

$$O_{b,x,y,c} = \gamma_c \frac{I_{b,x,y,c} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c.$$

Here, $\mu_c = \frac{1}{Z} \sum_{b,x,y} I_{b,x,y,c}$ denotes the per-channel mean, and $\sigma_c^2 = \frac{1}{Z} \sum_{b,x,y} I_{b,x,y,c}^2 - \mu_c^2$ denotes the per-channel variance of the inputs, and Z denotes the normalization constant summed over the minibatch b and spatial dimensions x and y. A small constant ϵ is added to the variance for numerical stability. The "scale" and "shift" parameters, γ_c and β_c respectively, are learnt during training. Running averages of the mean μ_c and variance σ_c^2 are also maintained during training, and these averages are used at test time to ensure the predictions are independent of other examples in the batch. For distributed training, the batch statistics are usually estimated locally on a subset of the training minibatch ("ghost batch normalization" (Hoffer et al., 2017)).

We discussed some of the benefits of batch normalization in the introduction. However batch normalization also has many limitations. It breaks the independence between training samples in a minibatch, which makes it hard to apply in certain models (Girshick, 2015), and also contradicts the assumptions of most theoretical models of optimization (Mandt et al., 2017; Park et al., 2019). The normalization operation itself is expensive, and can constitute a significant fraction of the total cost of computing a parameter update (Wu et al., 2018). It also performs poorly when the batch size is too small (Ioffe, 2017; Wu & He, 2018), which can limit the size of model that can be trained on a single device. There is therefore great interest in understanding the benefits of batch normalization and identifying simple alternatives.

Related Work. Balduzzi et al. (2017) and Yang et al. (2019) both argued that ResNets with identity skip connections and batch normalization layers on the residual branch preserve

I've never really looked at it this way, quite neat.

correlations between different minibatches in very deep networks, and Balduzzi et al. (2017) suggested this effect can be mimicked by initializing networks close to linear functions. However, neither paper gives a clear explanation of why batch normalization has this benefit. Furthermore, even deep linear networks are difficult to train with Gaussian weights (Saxe et al., 2013), which suggests that linearity is not sufficient. Veit et al. (2016) argued that residual networks can be interpreted as an ensemble over many paths of different depths, and they found empirically that this ensemble is dominated by short paths in normalized networks. However they do not explain why this occurs or discuss whether batch normalization would affect this conclusion.

Indeed, most papers studying the benefits of batch normalization have not discussed the combination of normalization and skip connections. Some authors have observed that batch normalization has a regularizing effect (Hoffer et al., 2017; Luo et al., 2019). Meanwhile, Santurkar et al. (2018) and Bjorck et al. (2018) both argued that the primary benefit of batch normalization is to improve the conditioning of the loss landscape, which enables stable training with larger learning rates. However while batch normalization does improve the conditioning of a network (Jacot et al., 2019), this conditioning decays rapidly as the network depth increases if skip connections are not present (Yang et al., 2019). Deep normalized networks without skip connections are therefore not trainable (Yang et al., 2019; Sankararaman et al., 2019).

Fixup. Zhang et al. (2019) proposed Fixup initialization, which can train deep residual networks without batch normalization. They also confirmed that Fixup can replace layer normalization in transformers (Vaswani et al., 2017) for machine translation. Fixup has four components:

1. The classification layer and final convolution of each residual branch are initialized to zero.
2. The initial weights of the remaining convolutions are scaled down by $d^{-1/(2m-2)}$, for a number of residual branches d and convolutions per branch m.
3. A scalar multiplier is introduced at the end of each residual branch, intialized to one.
4. Scalar biases are introduced before every convolution, linear or activation function layer, initialized to zero.

The authors claim that component 2 above is crucial, however we will demonstrate below that it is not necessary in practice at typical batch sizes. We show that a significantly simpler initialization scheme (SkipInit) enables efficient training of deep unnormalized ResNets so long as the batch size is not too big. The additional components of Fixup enable it to scale efficiently to slightly larger batch sizes than SkipInit, however neither Fixup or SkipInit scale to the very large batch sizes possible with batch normalization.

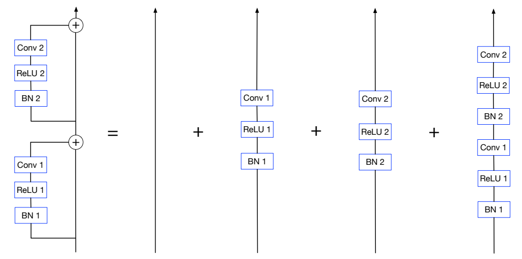


Figure 2. We can unroll a sequence of two residual blocks as an ensemble over 4 paths with different depths and shared weights.

3. A simple explanation of why deep normalized residual networks are trainable

Veit et al. (2016) argued that a residual network composed of d residual blocks can be described as an ensemble over 2^d paths, which can be grouped according to the number of residual blocks that they traverse (which we refer to as the depth of the path), with d being the total network depth):

$$\hat{f} = \prod_{i=1}^{d} \left(\hat{I} + \hat{f}_i\right) \tag{1}$$

$$= \hat{I} + \sum_i \hat{f}_i + \sum_{i>j} \hat{f}_i \hat{f}_j + ... + \prod_{i=1}^{d} \hat{f}_i. \tag{2}$$

In equation 1, $\hat{f}$ is an operator representing d residual blocks, $\hat{f}_i$ represents the i^{th} residual branch, and $\hat{I}$ is the identity operator representing the skip connection. We provide an illustration of this in figure 2. The distribution of path depths follows a binomial distribution, and the mean depth of a path is $d/2$. If the trainable depth were governed by the depth of a typical path, then introducing skip connections should roughly double the trainable depth. This was observed in unnormalized residual networks by Sankararaman et al. (2019). However, normalized residual networks can be trained for depths significantly deeper than twice the depth of their non-residual counterparts. To understand this effect, we consider the variance of hidden activations at initialization on both the skip path and the residual branch. To present our main argument, we focus here on the variance across multiple channels, but we also discuss the variance across the batch on a single channel in appendix A. For simplicity, we assume the convolutions are initialized using He initialization (He et al., 2015) to preserve the variance of incoming activations with ReLU non-linearities. We conclude:

Unnormalized networks: If the residual branch is not normalized, then we expect $\hat{I}$ and $\hat{f}_i$ to preserve the variance of their inputs. If the inputs are mean centered with variance 1, the skip path after i residual blocks will have expected variance 2^i. One can prevent the variance from exploding by introducing a factor of $\frac{1}{\sqrt{2}}$ at the end of each residual block. Notice however that all paths contribute equally, which im-

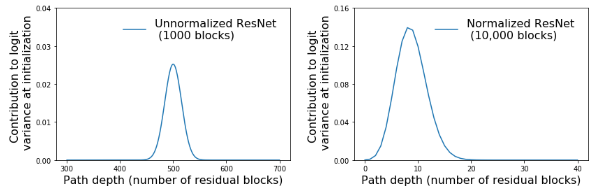


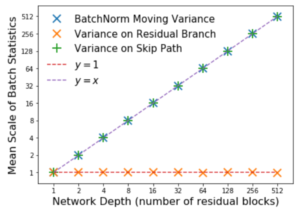
Figure 3. We simulate how paths of different depths contribute to the variance of the logits at initialization (see appendix B).

plies that the function at initialization is dominated by deep paths that traverse roughly half the residual branches.

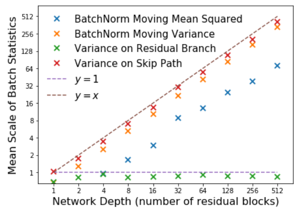
Normalized networks: When the residual branch is normalized, it is reasonable to assume that the output of $\hat{f}_i$ will have variance 1. Consequently, each residual block increases the variance by 1, and the activations just before the i^{th} residual block will have expected variance i. Therefore, in expectation, the variance of any path which traverses the i^{th} residual branch will be suppressed by a factor of i, which implies that the hidden activations are suppressed by a factor of $\sqrt{i}$. As shown in figure 3, this downscaling factor is sufficiently strong to ensure that 97% of the variance of a network with 10000 residual blocks will arise from shallow paths that traverse 15 or fewer residual branches. The depth of a typical residual block is proportional to the total number of residual blocks d, which suggests that batch normalization downscales residual branches by a factor on the order of $\sqrt{d}$. Although weaker than the factor of d proposed by Hanin & Rolnick (2018), we will find empirically that it is sufficiently strong to train networks with 1000 layers.

To verify this argument, we evaluate the variance across channels, as well as the batch normalization statistics, of two normalized residual networks at initialization in figure 4. We define both networks in appendix A. In figure 4(a), the variance on the skip path of a deep linear ResNet is approximately equal to the current depth i, while the variance at the end of each residual branch is approximately 1. This occurs because the batch normalization moving variance is also approximately equal to depth, confirming that normalization downscales the residual branch by a factor of $\sqrt{i}$. In figure 4(b), we consider a convolutional ResNet with ReLU activations evaluated on CIFAR-10. The variance on the skip path remains proportional to depth, but with a coefficient slightly below 1. This is likely caused by zero padding at the image boundary. The batch normalization moving variance is also proportional to depth, but slightly smaller than the variance across channels on the skip path. This occurs because ReLU activations introduce correlations between independent examples in the batch, which we discuss in appendix A. These correlations also cause the square of the batch normalization moving mean to grow with depth.

The above provides a simple explanation of why deep normalized ResNets are trainable. Our argument extends to



(a) Deep linear ResNet



(b) Deep ReLU ResNet

Figure 4. The dependence of the batch statistics at initialization on the depth of the residual block. a) A fully connected ResNet with linear activations and one normalization layer per residual branch, evaluated on random Gaussian inputs. The squared BatchNorm moving mean is close to zero (not shown). b) A convolutional ResNet with ReLU activations, evaluated on CIFAR-10.

other normalization variants and model architectures, including layer normalization (Ba et al., 2016) and "pre-norm" transformers (where the normalization layers are on the residual branch) (Radford et al., 2019). However our argument doesn't apply to the original transformer, which placed normalization layers on the skip path (Vaswani et al., 2017). This original transformer is famously difficult to train.

4. SkipInit; an alternative to normalization

We claim that normalization enables us to train deep residual networks, because in expectation it downscales the residual branch at initialization by a normalizing factor proportional to the square root of the network depth. To verify this claim, we propose a simple alternative to normalization, "SkipInit":

SkipInit: Put a scalar multiplier at the end of every residual branch, and initialize each multiplier to α (see figure 1).

After normalization is removed, it should be possible to implement SkipInit as a one line code change. In section 5.1, we show that we can train very deep residual networks, so long as α is initialized at a value of $(1/\sqrt{d})$ or smaller, where d denotes the total number of residual blocks (see table 1). We recommend setting $\alpha = 0$, so that the residual block represents the identity function at initialization. We emphasize that SkipInit is designed for ResNets which contain an identity skip connection (He et al., 2016b). We discuss how to extend SkipInit to the original ResNet-V1 formulation of ResNets in appendix E (He et al., 2016a).

We introduced Fixup in section 2 (Zhang et al., 2019), which also ensures that the residual block represents the identity at initialization. However Fixup contains multiple additional components. In practice, we have found that either component 1 or component 2 of Fixup is sufficient to train deep ResNet-V2s without normalization. Component 1 initializes residual blocks to the identity, while component 2 multiplies the residual branch by a factor $\beta \le (1/\sqrt{d})$. Component 3 enhances the rate of convergence, while component 4 is required for deep ResNet-V1s (He et al., 2016a) (see appendix

Table 1. Batch normalization enables us to train very deep residual networks. We can recover this benefit without normalization if we introduce a scalar multiplier α on the end of the residual branch and we initialize $\alpha = (1/\sqrt{d})$ or smaller (d denotes the number of residual blocks). In practice, we advocate initializing $\alpha = 0$. We also provide the optimal learning rates with error bars. In all three cases, the optimal learning rate is almost independent of depth.

Batch Normalization

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	93.5 ± 0.1	2^{-1} (2^{-1} to 2^{-1})
100	94.7 ± 0.1	2^{-1} (2^{-2} to 2^{-0})
1000	94.6 ± 0.1	2^{-2} (2^{-3} to 2^{-0})

SkipInit ($\alpha = 1/\sqrt{d}$)

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	93.0 ± 0.1	2^{-2} (2^{-2} to 2^{-1})
100	94.2 ± 0.1	2^{-1} (2^{-2} to 2^{-1})
1000	94.2 ± 0.0	2^{-1} (2^{-2} to 2^{-1})

SkipInit ($\alpha = 0$)

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	93.3 ± 0.1	2^{-2} (2^{-2} to 2^{-2})
100	94.2 ± 0.1	2^{-2} (2^{-2} to 2^{-2})
1000	94.3 ± 0.2	2^{-2} (2^{-3} to 2^{-1})

Table 2. We cannot train deep residual networks with SkipInit if $\alpha = 1$. Additionally, although it is helpful to divide the hidden activations of unnormalized ResNets by $\sqrt{2}$ after each residual block, this is only effective for networks containing a few hundred layers. Finally, we verify that L2 regularization is not required when training very deep residual networks with SkipInit ($\alpha = 0$).

SkipInit ($\alpha = 1$)

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	93.0 ± 0.1	2^{-2} (2^{-2} to 2^{-1})
100	–	–
1000	–	–

Divide residual block by $\sqrt{2}$

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	92.4 ± 0.1	2^{-2} (2^{-2} to 2^{-1})
100	88.9 ± 0.4	2^{-5} (2^{-5} to 2^{-5})
1000	–	–

SkipInit without L2 ($\alpha = 0$)

Depth	**Test accuracy (%)**	**Optimal learning rate**
16	89.8 ± 0.2	2^{-3} (2^{-3} to 2^{-3})
100	91.7 ± 0.2	2^{-2} (2^{-2} to 2^{-2})
1000	92.1 ± 0.1	2^{-2} (2^{-2} to 2^{-2})

E). We introduce SkipInit to clarify the minimal conditions required to train deep ResNets without normalization.

Both SkipInit and Fixup enable us to increase the depth of residual networks without increasing network width. By contrast, for Gaussian initialization, the trainable depth of vanilla networks without skip connections is proportional to width (Hanin & Rolnick, 2018; Hu et al., 2020).

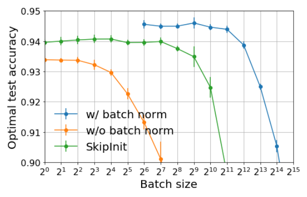
5. An empirical study of the benefits of batch normalization in residual networks

In this section, we provide a thorough empirical study of batch normalization in residual networks, which identifies multiple distinct benefits. In section 5.1, we verify the claims made in sections 3 and 4 by studying the minimal components required to train very deep residual networks. In section 5.2, we investigate the claim made in multiple previous papers (Bjorck et al., 2018; Santurkar et al., 2018) that the primary benefit of batch normalization is to increase the largest stable learning rate. We study the additional regularization benefits of batch normalization in section 5.3.

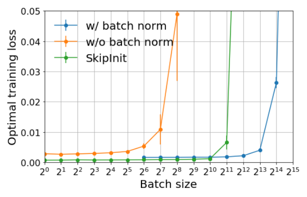
5.1. Normalization and deep networks

In table 1, we provide the mean performance of a n-2 Wide-ResNet (Zagoruyko & Komodakis, 2016), trained on CIFAR-10 for 200 epochs at batch size 64 at a range of depths n between 16 and 1000 layers. At each depth, we train the network 7 times for a range of learning rates on a logarithmic grid, and we independently measure the mean and standard deviation of the test accuracy for the best 5 runs. This procedure ensures that our results are not corrupted by outliers or failed runs. Throughout this paper, we train using SGD with heavy ball momentum, and fix the momentum coefficient $m = 0.9$. The optimal test accuracy is the mean performance at the learning rate whose mean test accuracy was highest. We always verify that the optimal learning rates are not at the boundary of our grid search. Although we tune the learning rate on the test set, we emphasize that our goal is not to achieve state of the art results. Our goal is to compare the performance of different training procedures, and we apply the same experimental protocol in each case. We hold the learning rate constant for 100 epochs, before dropping the learning rate by a factor of 2 every 10 epochs. This simple schedule achieves higher test accuracies than the original schedule proposed in He et al. (2016a). We apply data augmentation including padding, random crops and left-right flips and we also use L2 regularization with a coefficient of 5×10^{-4}. We initialize convolutional layers using He initialization (He et al., 2015). We provide the optimal training losses in appendix C.

As expected, batch normalized Wide-ResNets are trainable for a wide range of depths, and the optimal learning rate is only weakly dependent on the depth. We can recover this effect without normalization by incorporating SkipInit and initializing $\alpha = (1/\sqrt{d})$ or smaller, where d denotes the number of residual blocks. This provides additional evi-

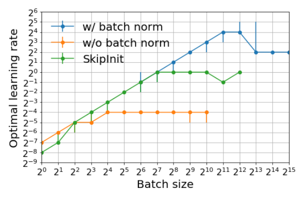


(a)

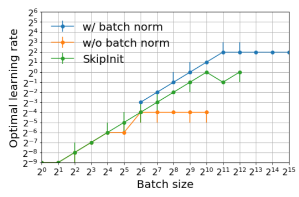


(b)

Figure 5. We achieve a higher test accuracy with batch normalization than without batch normalization, and we are also able to train efficiently at much larger batch sizes. SkipInit substantially reduces the gap in performance for small/moderate batch sizes, but it still significantly underperforms batch normalization when the batch size is large. SkipInit also achieves smaller training losses than batch normalization for batch sizes $B \lesssim 1024$. Results are for a 16-4 Wide-ResNet, trained on CIFAR-10 for 200 epochs at a range of batch sizes. We provide the test accuracy at the learning rate for which the test accuracy was maximized, and the training loss at the learning rate for which the training loss was minimized.



(a) (for test accuracy)



(b) (for training loss)

Figure 6. The optimal learning rates with and without batch normalization, as well as with SkipInit, are similar when using small batch sizes (especially when maximizing test accuracy). Batch normalization enables us to use larger learning rates (confirming that it has a conditioning benefit), but this is only useful when we scale training to larger batch sizes. We observe that the optimal learning rate is proportional to batch size for small/moderate batch sizes, but the learning rate is constant when the batch size is large.

dence to support our claim that batch normalization enables us to train deep residual networks by biasing the network towards shallow paths at initialization. Just like normalized networks, the optimal learning rate with SkipInit is almost independent of the network depth. SkipInit slightly underperforms batch normalization on the test set at all depths. However as shown in sections 5.3 and 6, it may be possible to close this gap with sufficiently strong regularization.

In table 2, we verify that one cannot train deep residual networks with SkipInit if $\alpha = 1$, confirming that it is necessary to downscale the residual branch. We also confirm that for unnormalized residual networks, it is not sufficient merely to ensure the activations do not explode on the forward pass (which can be achieved by simply multiplying the activations by $(1/\sqrt{2})$ every time the residual branch and skip path merge). Finally, we noticed that, at initialization, the loss in very deep networks is dominated by the L2 regularization term, which causes the weights to shrink rapidly early in training. To clarify whether this effect is necessary, we also evaluated the performance of SkipInit ($\alpha = 0$) without L2 regularization. We find that L2 regularization is not necessary for trainability. This demonstrates that we can train very deep residual networks without normalization and without reducing the scale of the weights at initialization.

5.2. Normalization and large batch training

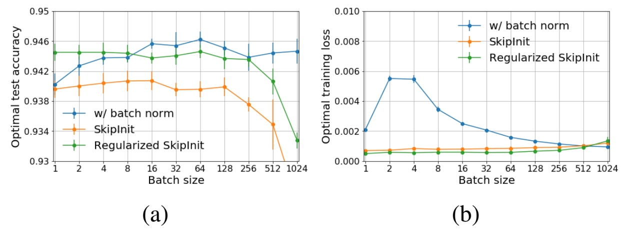
In this section, we investigate the claim made by Santurkar et al. (2018) and Bjorck et al. (2018) that the primary benefit of batch normalization is to improve the conditioning of the loss landscape, which increases the largest stable learning rate. To study this, in figure 5 we provide the mean performance of a 16-4 Wide-ResNet, trained on CIFAR-10 for 200 epochs at a wide range of batch sizes. We follow the same experimental protocol described in section 5.1, however we average over the best 12 out of 15 runs. To enable us to consider extremely large batch sizes on a single GPU, we evaluate the batch statistics over a fixed ghost batch size of 64 (Hoffer et al., 2017), before accumulating gradients to form larger batches. We therefore are unable to consider batch sizes below 64 with batch normalization. Evaluating batch statistics over a fixed number of training examples improves the test accuracy achieved with large batches (Hoffer et al., 2017; Goyal et al., 2017) and reduces communication overheads in distributed training. We repeat this experiment in the small batch limit in section 5.3, evaluating batch statistics over the full training batch.

We see that performance is better with batch normalization than without batch normalization on both the test set and the training set at all batch sizes. For clarity, in figure 5(b) we provide the training loss excluding the L2 regularization term.[1] Additionally, both with and without batch normalization, the final test accuracy is independent of batch size in the small batch limit, before beginning to decrease when the batch size exceeds some threshold.[2] This threshold is significantly larger when batch normalization is used, which demonstrates that one can efficiently scale training to larger batch sizes in normalized networks (Goyal et al., 2017).

To better understand why batch normalized networks can scale training efficiently to larger batches, we provide the optimal learning rates which maximize the test accuracy and minimize the training loss in figure 6. When the batch size is small, the optimal learning rates are proportional to the batch size (Mandt et al., 2017; Smith & Le, 2017), and the optimal learning rate is similar with and without batch normalization. However when the batch size is large, the optimal learning rates are independent of batch size (McCan-

[1]Normalized networks achieve smaller L2 losses because they can shrink the weights without changing the network function.

[2]As batch size grows, the number of updates decreases since the number of epochs is fixed. The performance might not degrade with batch size under a constant step budget (Shallue et al., 2018).



?

Figure 7. Batch normalization has a regularization effect, whereby the training loss falls as the batch size increases, but the test accuracy is maximized for an intermediate batch size, $B \approx 64$. Regularized SkipInit outperforms batch normalization on the test set for very small batch sizes. We train without ghost batch normalization, evaluating batch statistics over the full training batch. We use a drop probability of 0.6 when Dropout is used.

dlish et al., 2018). Intuitively, we have reached the largest stable learning rate, above which training will diverge. We find that batch normalization increases the largest stable learning rate, confirming that it improves the conditioning of the loss (Santurkar et al., 2018; Bjorck et al., 2018). It is this benefit which enables us to efficiently scale training to larger batches. SkipInit reduces the gap in test accuracy between normalized and unnormalized networks, and it achieves smaller training losses than batch normalization when the batch size is small ($B \lesssim 1024$). However SkipInit does not share the full conditioning benefits of batch normalization, and therefore it is not competitive with batch normalized networks in the large batch limit. We show in section 6 that similar limitations apply to Fixup.

Our results confirm that batch normalization increases the largest stable learning rate, which enables large batch training. However we emphasize that this benefit does not increase the test accuracy one can achieve within a finite epoch budget. As figures 5 and 6 demonstrate, we always achieve the best performance at small batch sizes for which the optimal learning rates with and without batch normalization are significantly smaller than the largest stable learning rate. Santurkar et al. (2018) and Bjorck et al. (2018) claimed that the primary benefit of batch normalization is to increase the largest stable learning rate, however our results show that this is not correct for ResNets. In ResNets, the primary benefit of batch normalization is to bias the network towards shallow paths. This allows us to train deeper networks, and it also improves the test accuracies of shallow networks. We show in the next two sections that the gap in test accuracy between batch normalization and SkipInit can be further reduced or completely closed with additional regularization.

5.3. Normalization and regularization

In this section, we study the regularization benefit of batch normalization at a range of batch sizes. We also explore ways to recover this regularization benefit using "Regularized SkipInit", which is comprised of three components:

Table 3. When training ResNet50-V2 on ImageNet, both SkipInit and Fixup are competitive with batch normalization for small or moderate batch sizes, while batch normalization performs best when the batch size is very large. Both SkipInit and Fixup achieve higher validation accuracies than batch normalization with extra regularization for small batch sizes. We train for 90 epochs and perform a grid search to identify the optimal learning rate which maximizes the top-1 validation accuracy. We perform a single run at each learning rate and report both top-1 and top-5 accuracy scores. We use a drop probability of 0.2 when Dropout is used.

	Batch size		
Test accuracy:	256	1024	4096
Batch normalization	75.0/92.2	74.9/92.1	74.9/91.9
Fixup	74.8/91.8	74.6/91.7	73.0/90.6
SkipInit + Biases	74.9/91.9	74.6/91.8	70.8/89.2
Fixup + Dropout	75.8/92.5	75.6/92.5	74.8/91.8
Regularized SkipInit	75.6/92.4	75.5/92.5	72.7/90.7

1. We introduce a scalar multiplier at the end of each residual branch, initialized to zero (SkipInit).
2. We introduce biases to the convolutional layers.
3. We apply Dropout on the classification layer.

We provide the performance of our 16-4 Wide-ResNet at a range of batch sizes in the small batch limit in figure 7. To focus on the regularization benefit of batch normalization, we evaluate the batch statistics over the full training batch, enabling us to consider any batch size $B \geq 1$ (note that batch normalization reduces to instance normalization when $B = 1$). We provide the corresponding optimal learning rates in appendix D. The test accuracy achieved with batch normalization initially improves as the batch size rises, before decaying for batch sizes $B \gtrsim 64$. Meanwhile, the training loss increases as the batch size rises from 1 to 2, but then decreases consistently as the batch size rises further. This confirms that noise arising from uncertainty in the batch statistics does have a generalization benefit if properly tuned (Luo et al., 2019), which is why we use a ghost batch size of 64 in preceding sections. The performance of SkipInit and Regularized SkipInit are independent of batch size in the small batch limit, and consequently Regularized SkipInit achieves higher test accuracies than batch normalization when the batch size is very small. Note that we introduced Dropout (Srivastava et al., 2014) to show that extra regularization may be necessary to close the performance gap when normalization is removed, but more sophisticated regularizers would likely achieve higher test accuracies.

?

6. A comparison of batch normalization, SkipInit and Fixup on ImageNet

We confirm SkipInit scales to large challenging data distributions by providing an empirical comparison of SkipInit, Fixup initialization (Zhang et al., 2019) and batch normalization on ImageNet. Since SkipInit is designed for residual networks with an identity skip connection, we consider the ResNet50-V2 architecture (He et al., 2016b). We provide additional experiments on ResNet50-V1 in appendix E (He et al., 2016a). We use the original architectures, and do not apply the popular modifications to these networks described in Goyal et al. (2017). When batch normalization is used, we set the ghost batch size equal to 256. We train for 90 epochs. The learning rate is linearly increased from 0 to the specified value over the first 5 epochs of training (Goyal et al., 2017), and then held constant for 40 epochs, before decaying it by a factor of 2 every 5 epochs. As before, we tune the learning rate at each batch size on a logarithmic grid. We provide the optimal validation accuracies in table 3. We found that including biases in convolutional layers led to a small boost in validation accuracy for SkipInit, and we therefore included biases in all SkipInit runs. SkipInit matches the validation set performance of batch normalization and Fixup at the standard batch size of 256. However both SkipInit and Fixup underperform batch normalization when the batch size is very large. Both SkipInit and Fixup can outperform batch normalization on the validation set with extra regularization (Dropout) for small batch sizes.

Fixup outperforms SkipInit when the batch size is very large, suggesting that the second component of Fixup has a small conditioning benefit (see section 2). However we found in section 5.1 that this component is not necessary to train very deep residual networks, and we confirm here that it does not improve performance at the standard batch size of 256.

7. Discussion

Our work confirms that batch normalization has three principal benefits. In (subjective) order of importance,

1. Batch normalization can train very deep ResNets.
2. Batch normalization improves conditioning, which enables us to scale training to larger batch sizes.
3. Batch normalization has a regularizing effect.

This work identifies a simple explanation for benefit 1: normalization layers bias deep residual networks towards shallow paths. These shallow paths have well-behaved gradients, enabling efficient training (Balduzzi et al., 2017; Hanin & Rolnick, 2018; Yang et al., 2019). This benefit also applies to other normalization schemes, including layer normalization in "pre-norm" transformers (Radford et al., 2019). A single normalization layer per residual branch is sufficient, and normalization layers should not be placed on the skip path (as in the original transformer of Vaswani et al. (2017)). We can recover benefit 1 without normalization by introducing a scalar multiplier on the residual branch inversely proportional to the square root of the network depth (or zero for simplicity). This one line code change can train deep residual networks without normalization, and also enhances the performance of shallow residual networks. We therefore conclude that one no longer needs normalization layers to train deep residual networks with small batch sizes (e.g., batch sizes below 1024 for ResNet-50 on ImageNet).

However, the conditioning benefit (benefit 2) of batch normalization remains important when one wishes to train with large batch sizes. This could make normalization necessary in time-critical situations, for instance if a production model is retrained frequently in response to changing user preferences. Also, since batch normalization has a regularizing effect (benefit 3), it may be beneficial in some architectures if one wishes to achieve the highest possible test accuracy. Note however that one can often exceed the test accuracy of normalized networks by introducing alternative regularizers (see section 6 or Zhang et al. (2019)). We therefore believe future work should focus on identifying an alternative to batch normalization that recovers its conditioning benefits.

We would like to comment briefly on the similarity between SkipInit for residual networks, and Orthogonal initialization of vanilla fully connected tanh networks (Saxe et al., 2013). Orthogonal initialization is currently the only initialization scheme capable of training very deep networks without skip connections. It initializes the weights of each layer as an orthogonal matrix, such that the activations after a linear layer are a rotation (or reflection) of the activations before the layer. Meanwhile, the tanh non-linearity is approximately equal to the identity for small activations over a region of scale 1 around the origin. Intuitively, if the incoming activations are mean centered with scale 1, they will pass through the non-linearity almost unchanged. Since rotations compose, the approximate action of the entire network at initialization is to rotate (or reflect) the input. Like SkipInit, the initial functions generated by orthogonal initialization of vanilla tanh networks are almost independent of the network depth. However ReLUs are not compatible with orthogonal initialization, since they are not linear about the origin. This has limited the use of orthogonal initialization in practice.

To conclude: Batch normalization and SkipInit have one crucial property in common. At initialization, they both bias deep residual networks towards an ensemble of shallow paths with well-behaved gradients. This property enables us to train deep residual networks without increasing the network width (Hanin & Rolnick, 2018; Hu et al., 2020), and it is therefore a major factor behind the popularity of normalized residual networks in deep learning. Batch normalization also has both a conditioning benefit and a regularization benefit. However, we demonstrate in this paper that one can train competitive networks without normalization by choosing a sensible initialization scheme, introducing extra regularization, and training with small minibatches.

Acknowledgements

We thank Jeff Donahue, Chris Maddison, Erich Elsen, James Martens, Razvan Pascanu, Chongli Qin, Karen Simonyan, Yann Dauphin, Esme Sutherland and Yee Whye Teh for various discussions that have helped improve the paper.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 342–350. JMLR. org, 2017.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pp. 7694–7705, 2018.

Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hanin, B. and Rolnick, D. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pp. 571–581, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. 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.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Hu, W., Xiao, L., and Pennington, J. Provable benefit of orthogonal initialization in optimizing deep linear networks. *arXiv preprint arXiv:2001.05992*, 2020.

Ioffe, S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in neural information processing systems*, pp. 1945–1953, 2017.

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

Jacot, A., Gabriel, F., and Hongler, C. Freeze and chaos for dnns: an ntk view of batch normalization, checkerboard and boundary effects. *arXiv preprint arXiv:1907.05715*, 2019.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Luo, P., Wang, X., Shao, W., and Peng, Z. Towards understanding regularization in batch normalization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJlLKjR9FQ`.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Park, D. S., Sohl-Dickstein, J., Le, Q. V., and Smith, S. L. The effect of network width on stochastic gradient descent and generalization: an empirical study. *arXiv preprint arXiv:1905.03776*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

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

Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *arXiv preprint arXiv:1904.06963*, 2019.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. *arXiv preprint arXiv:1911.09737*, 2019.

Smith, S. L. and Le, Q. V. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veit, A., Wilber, M. J., and Belongie, S. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pp. 550–558, 2016.

Wu, S., Li, G., Deng, L., Liu, L., Wu, D., Xie, Y., and Shi, L. L1-norm batch normalization for efficient training of deep neural networks. *IEEE transactions on neural networks and learning systems*, 2018.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. S., and Pennington, J. 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.

Xie, Q., Hovy, E., Luong, M.-T., and Le, Q. V. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

A. The influence of ReLU non-linearities on the batch statistics in residual networks.

In figure 4 of the main text, we studied the batch statistics of residual blocks at a wide range of depths in two different architectures; a deep linear fully connected normalized ResNet and a deep convolutional normalized ResNet with ReLU non-linearities. We now define both models in full:

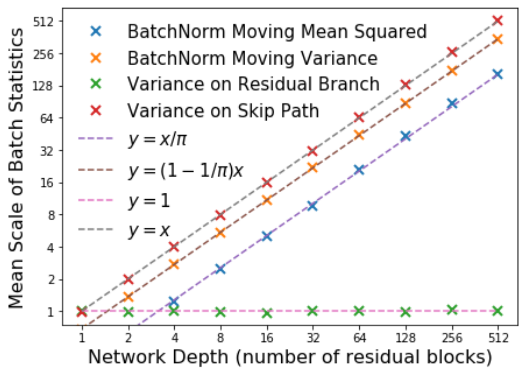
Deep fully connected linear ResNet: The inputs are 100 dimensional vectors composed of independent random samples from the unit normal distribution, and the batch size is 1000. These inputs first pass through a batch normalization layer and a single fully connected linear layer of width 1000. We then apply a series of residual blocks. Each block contains a skip path (the identity), and a residual branch composed of a batch normalization layer and a fully connected linear layer of width 1000. All linear layers are initialized with LeCun normal initialization (LeCun et al., 2012) to preserve variance in the absence of non-linearities.

Deep convolutional ReLU ResNet: The inputs are batches of 100 images from the CIFAR-10 training set. We first apply a convolution of width 100 and stride 2, followed by a batch normalization layer, a ReLU non-linearity, and an additional convolution of width 100 and stride 2. We then apply a series of residual blocks. Each block contains a skip path (the identity), and a residual branch composed of a batch normalization layer, a ReLU non-linearity, and a convolution of width 100 and stride 1. All convolutions are initialized with He initialization (He et al., 2015) to preserve variance in the presence of ReLU non-linearities.

In both networks, we evaluate the variance across channels at initialization on both the skip path and at the end of the residual branch, as well as the mean moving variance (i.e., the single channel moving variance averaged over channels) and mean squared moving mean (i.e., squared moving mean averaged over channels) of the batch normalization layer. To obtain the batch normalization statistics, we set the momentum parameter of the batch normalization layers to 0, and then apply a single update to the batch statistics.

In the main text, we found that for the deep linear network, the variance across channels on the skip path was equal to the mean moving variance of the batch normalization layer, while the mean squared moving mean of the batch normalization layer was close to zero. However when we introduced ReLU non-linearities in the deep convolutional ResNet, the mean moving variance of the batch normalization layer was smaller than the variance across channels on the skip path, and the mean squared moving mean of the normalization layer grew proportional to the depth.

To clarify the origin of this effect, we consider an additional fully connected deep ReLU ResNet in figure 8. We form this network from the fully connected linear ResNet above by inserting a ReLU layer after each batch normalization layer, and we replace LeCun initialization with He initialization. This network is easier to analyze than the ConvNet since the inputs are drawn from the normal distribution and there are no boundary effects due to padding.



(a) Fully connected deep ReLU ResNet

Figure 8. The batch statistics at initialization of a normalized deep fully connected network with ReLU non-linearities, evaluated on random inputs drawn from a Gaussian distribution.

In figure 8, we find that the variance over channels on the skip path is approximately equal to the depth of the residual block d, while the variance over channels at the end of the residual branch is approximately 1. This is identical to the deep linear network and matches our theoretical predictions in section 3. However, the mean moving variance of the batch normalization layer is approximately equal to $d(1 - 1/\pi)$, while the mean squared moving mean of the normalization layer is approximately equal to d/π. Notice that the combination of the mean squared moving mean and the mean moving variance of the normalization layer is equal to the depth of the block, which confirms that the batch normalization layer still reduces the variance over channels on the residual branch by a factor of depth d.

To understand this plot, we note that the outputs of a ReLU non-linearity have non-zero mean, and therefore the ReLU layer will cause the hidden activations of different examples in the batch to become increasingly correlated. Because the hidden activations of different examples are correlated, the variance across channels (the variance of hidden activations across multiple examples and multiple channels) becomes different from the variance over a single channel (the variance across multiple examples for a single channel).

For example, consider a simple network whose inputs x are a batch of B samples of dimension W from a Gaussian distribution with mean $\mathbf{E}(x_{ij}) = 0$ and covariance $\mathbf{E}(x_{ik}x_{jl}) = \delta_{ij}\delta_{kl}$, where δ_{ij} is the dirac delta function. The first dimension corresponds to the features and the second dimension corresponds to the batch. The network consists of a ReLU layer, followed by a fully connected linear layer with output dimension H, and finally a batch normalization layer. The linear layer ω is initialized from an uncorrelated Gaussian distribution with mean $\mathbf{E}(\omega_{ij}) = 0$ and covariance $\mathbf{E}(\omega_{ij}\omega_{kl}) = 2\,\delta_{ik}\delta_{jl}/W$ (He initialization).

The inputs to the normalization layer, $y_{ij} = \sum_k \omega_{ik} x^+_{kj}$, where x^+ denotes the output of the ReLU non-linearity. The mean activation $\mathbf{E}(y_{ij}) = 0$, while the covariance,

$$\begin{aligned}\mathbf{E}(y_{ij}y_{lm}) &= \mathbf{E}\left(\sum_{kn} \omega_{ik}x^+_{kj}\omega_{ln}x^+_{nm}\right) \\ &= \frac{2\delta_{il}\sum_k \mathbf{E}\left(x^+_{kj}x^+_{km}\right)}{W} \\ &= \delta_{il}\left(\frac{1+(\pi-1)\delta_{jm}}{\pi}\right). \qquad (3)\end{aligned}$$

To derive equation 3, we recalled that $x^+_{ij} = max(0, x_{ij})$, which implies that $\mathbf{E}(x^+_{kj}) = \sqrt{1/2\pi}$ while $\mathbf{E}\left((x^+_{kj})^2\right) = 1/2$. We can now apply equation 3 to compute the expected variance across multiple channels,

$$\begin{aligned}\mathbf{E}(\sigma^2) &= \mathbf{E}(y^2_{ij}) - \mathbf{E}(y_{ij})^2 \\ &= 1.\end{aligned}$$

The expected mean squared activation across a single channel (the expected BatchNorm moving mean squared),

$$\begin{aligned}\mathbf{E}\left(\mu^2_c\right) &= \mathbf{E}\left(\left(\frac{1}{B}\sum_j y_{cj}\right)^2\right) \\ &= \frac{1}{B^2}\sum_{jk}\mathbf{E}\left(y_{cj}y_{ck}\right) \\ &= \frac{1}{\pi} + \frac{\pi-1}{\pi B} \\ &\approx 1/\pi. \qquad (4)\end{aligned}$$

Note that to reach equation 4 we assumed that $B \gg 1$. It is immediately clear that the expected variance across a single channel (the expected BatchNorm moving variance),

$$\mathbf{E}\left(\sigma^2_c\right) = \mathbf{E}\left(\frac{1}{B}\sum_j y^2_{cj}\right) - \mathbf{E}\left(\mu^2_c\right) \approx (1 - 1/\pi).$$

These predictions match the scaling factors for batch normalization statistics we observed empirically in figure 8. We emphasize that this derivation does not directly apply to the residual network, since it does not account for the presence of paths containing multiple normalization layers. However it does provide a simple example in which ReLU non-linearities introduce correlations in the hidden activations between training examples. These correlations are responsible for the emergence of a large drift in the mean value of each hidden activation as the network depth increases, and a corresponding reduction in the variance observed on a single channel across multiple examples.

Table 4. The training losses, and associated optimal learning rates, of an n-2 Wide-ResNet at a range of depths n. We train on CIFAR-10 for 200 epochs with either batch normalization or SkipInit.

Batch Normalization

Depth	Training loss	Optimal learning rate
16	0.007 ± 0.000	2^{-2} (2^{-2} to 2^{-2})
100	0.001 ± 0.000	2^{-3} (2^{-3} to 2^{-2})
1000	0.001 ± 0.000	2^{-3} (2^{-4} to 2^{-3})

SkipInit ($\alpha = 1/\sqrt{d}$)

Depth	Training loss	Optimal learning rate
16	0.004 ± 0.000	2^{-3} (2^{-3} to 2^{-3})
100	0.001 ± 0.000	2^{-4} (2^{-4} to 2^{-4})
1000	0.001 ± 0.000	2^{-3} (2^{-4} to 2^{-3})

SkipInit ($\alpha = 0$)

Depth	Training loss	Optimal learning rate
16	0.004 ± 0.000	2^{-3} (2^{-3} to 2^{-3})
100	0.001 ± 0.000	2^{-4} (2^{-4} to 2^{-3})
1000	0.001 ± 0.000	2^{-4} (2^{-4} to 2^{-4})

B. Estimating the contributions to logit variance from paths of different depths

In figure 3, we estimated the contributions to the variance of the logits at initialization from paths of different depths, for normalized and unnormalized residual networks. We compute these estimates using a simple dynamical program. We begin with 0 residual blocks, which has a single path of depth 0 with variance 1. We progressively add a single block one at a time, and we calculate the variance arising from paths of every possible path depth. When estimating the variance of paths in non-residual networks, we assume that every path has variance 1, such that we simply have to count the paths. Therefore, if $V_q(i)$ denotes the variance arising from all paths of depth q after i residual blocks, then $V_0(0) = 1$ and $V_q(i) = V_q(i-1) + V_{q-1}(i-1)$. However when estimating the variance of paths in normalized residual networks, we assume that every path which traverses the i^{th} residual block is suppressed by a factor of i, which implies that $V_q(i) = V_q(i-1) + V_{q-1}(i-1)/i$.

C. The optimal training losses of Wide-ResNets at a range of network depths

In tables 4 and 5, we provide the minimum training losses, as well as the optimal learning rates at which the training loss is minimized, when training an n-2 Wide-ResNet for a range of depths n. At each depth, we train for 200 epochs on CIFAR-10 following the training procedure described in section 5.1 of the main text. These results correspond to the same architectures considered in tables 1 and 2, where we provided the associated test set accuracies. We provide the

Table 5. The training losses, and associated optimal learning rates, of an n-2 Wide-ResNet at a range of depths n. We train on CIFAR-10 for 200 epochs. We train with SkipInit (for different values of α), or we introduce a factor of $1/\sqrt{2}$ after each residual block.

SkipInit ($\alpha = 1$)

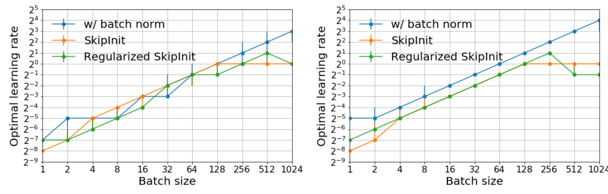
Depth	**Training loss**	**Optimal learning rate**
16	0.004 ± 0.000	2^{-3} (2^{-4} to 2^{-3})
100	–	–
1000	–	–

Divide residual block by $\sqrt{2}$

Depth	**Training loss**	**Optimal learning rate**
16	0.013 ± 0.000	2^{-3} (2^{-3} to 2^{-3})
100	0.066 ± 0.015	2^{-6} (2^{-6} to 2^{-6})
1000	–	–

SkipInit without L2 ($\alpha = 0$)

Depth	**Training loss**	**Optimal learning rate**
16	0.008 ± 0.000	2^{-3} (2^{-3} to 2^{-3})
100	0.001 ± 0.000	2^{-3} (2^{-4} to 2^{-2})
1000	0.000 ± 0.000	2^{-2} (2^{-2} to 2^{-2})



(a) (for test accuracy) (b) (for training loss)

Figure 9. The optimal learning rates of SkipInit, Regularized SkipInit and Batch Normalization, for a 16-4 Wide-ResNet trained for 200 epochs on CIFAR-10. We evaluate the batch statistics over the full training batch. All three methods have similar optimal learning rates in the small batch limit (proportional to batch size).

training loss excluding the L2 regularization term (i.e. the training set cross entropy), since one cannot meaningfully compare the L2 regularization penalty of normalized and unnormalized networks. In table 4, we confirm that both batch normalization and SkipInit can achieve training losses which depend only weakly on the network depth.

In table 5, we confirm that SkipInit cannot train very deep residual networks if the initial value of the scalar multipliers is too large. We also confirm that one cannot train very deep residual networks solely by normalizing the forward pass (which can be achieved by dividing the output of each residual block by $\sqrt{2}$). Finally, we confirm that SkipInit can achieve extremely small training losses across a wide range of depths even if we do not apply L2 regularization.

Table 6. We train ResNet50-V1 on ImageNet for 90 epochs. Fixup performs well when the batch size is small, but performs poorly when the batch size is large. Regularized SkipInit performs poorly at all batch sizes, but its performance improves considerably if we add a scalar bias before the final ReLU in each residual block (after the skip connection and residual branch merge). We perform a grid search to identify the optimal learning rate which maximizes the top-1 validation accuracy. We perform a single run at each learning rate and report both top-1 and top-5 accuracy scores. We use a drop probability of 0.2 for Regularized SkipInit. We emphasize that ResNet-V1 does not have an identity skip connection.

	Batch size		
Test accuracy:	256	1024	4096
Batch normalization	75.6/92.5	75.3/92.4	75.4/92.4
Fixup	74.4/91.6	74.4/91.7	72.4/90.3
Regularized SkipInit	70.0/89.2	68.4/87.8	68.2/87.9
Regularized SkipInit + Scalar Bias	75.2/92.4	74.9/92.0	70.8/89.6

D. The optimal learning rates of Wide-ResNets in the small batch limit

In figure 9, we provide the optimal learning rates of SkipInit, Regularized SkipInit and Batch Normalization, when training a 16-4 Wide-ResNet on CIFAR-10. These optimal learning rates correspond to the training losses and test accuracies provided in figure 7. We evaluate the batch statistics for batch normalization layers over the full training batch.

E. ImageNet results for ResNet-50-V1

In table 6, we present the performance of batch normalization, Fixup and Regularized SkipInit when training Resnet-50-V1 (He et al., 2016a). Unlike ResNet-V2 and Wide-ResNets, this network introduces a ReLU at the end of the residual block after the skip connection and residual branch merge. We find that Fixup slightly underperforms batch normalization when the batch size is small, but considerably underperforms batch normalization when the batch size is large (similar to the results on ResNet-50-V2). However, Regularized SkipInit significantly underperforms both batch normalization and Fixup at all batch sizes. This is not surprising, since we designed SkipInit for models which contain an identity skip connection through the residual block. We provide additional results for a modified version of Regularized SkipInit, which contains a single additional scalar bias in each residual block, just before the final ReLU (after the skip connection and residual branch merge). This scalar bias eliminates the gap in validation accuracy between Fixup and Regularized SkipInit when the batch size is small. We conclude that only two components of Fixup are essential to train the original ResNet-V1: initializing the residual branch at zero, and introducing a scalar bias after the skip connection and residual branch merge.