A ROBUST INITIALIZATION OF RESIDUAL BLOCKS FOR EFFECTIVE RESNET TRAINING WITHOUT BATCH NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Batch Normalization is an essential component of all state-of-the-art neural networks architectures. However, since it introduces many practical issues, much recent research has been devoted to designing normalization-free architectures. In this paper, we show that weights initialization is key to train ResNet-like normalization-free networks. In particular, we propose a slight modification to the summation operation of a block output to the skip-connection branch, so that the whole network is correctly initialized. We show that this modified architecture achieves competitive results on CIFAR-10 without further regularization nor algorithmic modifications.

1 INTRODUCTION

Batch normalization (Ioffe & Szegedy, 2015; Summers & Dinneen, 2020), in conjunction with skip connections (He et al., 2016a;b), has allowed the training of significantly deeper networks, so that most state-of-the-art architectures are based on these two paradigms.

The main reason why this combination works well is that it yields well behaved gradients (removing *mean-shift*, avoiding *vanishing* or *exploding* gradients). As a consequence, the training problem can be "easily" solved by SGD or other first-order stochastic optimization methods. Furthermore, batch normalization can have a regularizing effect (Hoffer et al., 2017; Luo et al., 2019).

However, while skip connections can be easily implemented and integrated in any network architecture without major drawbacks, batch normalization poses a few practical challenges. As already observed and discussed by Brock et al. (2020; 2021) and references therein, batch normalization adds a significant memory overhead, introduces a discrepancy between training and inference time, has a tricky implementation in distributed training, performs poorly with small batch sizes (Yan et al., 2020) and breaks the independence between training examples in a minibatch, which can be extremely harmful for some learning tasks (Lee et al., 2020; Lomonaco et al., 2020).

For these reasons a new stream of research emerged which aims at removing batch normalization from modern architectures. Several works (Zhang et al., 2019; De & Smith, 2020; Bachlechner et al., 2020) aim at removing normalization layers by introducing a learnable scalar at the end of the residual branch, i.e., computing a residual block of the form $x_l = x_{l-1} + \alpha f(x_{l-1})$. The scalar α is often initialized to zero so that the gradient is dominated, early on in the training, by the skip path. While these approaches have been shown to allow the training of very deep networks, they still struggle to obtain state-of-the-art test results on challenging benchmarks.

More recently Brock et al. (2020; 2021) proposed an approach that combines a modification of the latter residual block with a careful initialization, a variation of the Scaled Weight Standardization (Huang et al., 2017; Qiao et al., 2020) and a novel adaptive gradient clipping technique. Such combination has been shown to obtain competitive results on challenging benchmarks.

In this work we propose a simple modification of the residual block summation operation that, together with a careful initialization, allows to train deep residual networks without any normalization layer. Such scheme does not require the use of any standardization layer nor algorithmic modification. Our contributions are as follows:

- We show that while *NFNets* of Brock et al. (2020; 2021) enjoy a perfect forward variance (as already noted by Brock et al. 2020), it puts the network in a regime of *exploding gradients*. This is shown by looking at the variance of the derivatives of the loss w.r.t. to the feature maps at different depths.
- We propose a simple modification of the residual layer and then develop a suitable initialization scheme building on the work of He et al. (2015).
- We show that the proposed architecture achieves competitive results on CIFAR-10 (Krizhevsky et al., 2009).

2 BACKGROUND

As highlighted in a number of recent studies (Hanin & Rolnick, 2018; Arpit et al., 2019; Dauphin & Schoenholz, 2019), weights initialization is crucial to make deep networks work in absence of batch normalization. In particular, the weights at the beginning of the training process should be set so as to correctly propagate the forward activation and the backward gradients signal in terms of mean and variance.

This kind of analysis was first proposed by Glorot & Bengio (2010) and later extended by He et al. (2015). These seminal studies considered architectures composed by a sequence of convolutions and Rectified Linear Units (ReLU), which mainly differ from modern ResNet architectures for the absence of skip-connections.

The analysis in He et al. (2015) investigates the variance of each response layer l (*forward variance*):

$$z_l = \text{ReLU}(x_{l-1}), \qquad x_l = W_l z_l.$$

The authors find that if $\mathbb{E}[x_{l-1}] = 0$ and $\text{Var}[x_{l-1}] = 1$ the output maintains zero mean and unit variance if we initialize the kernel matrix in such a way that:

$$\text{Var}[W] = \frac{2}{n_{\text{in}}}, \tag{1}$$

where $n_{\text{in}} = k^2 c$ with k the filter dimension and c the number of input channels (*fan in*).

A similar analysis is carried out considering the gradient of the loss w.r.t. each layer response (*backward variance*) $\frac{\partial \mathcal{L}}{\partial x_l}$. In this case we can preserve zero mean and constant variance if we have

$$\text{Var}[W] = \frac{2}{n_{\text{out}}}, \tag{2}$$

where $n_{\text{out}} = k^2 d$ with k the filter dimension and d the number of output channels (*fan out*).

Note that equations (1) and (2) only differ for a factor which, in most common network architectures, is in fact equal to 1 in the vast majority of layers. Therefore, the initialization proposed by He et al. (2015) should generally lead to the conservation of both *forward* and *backward* signals.

The two derivations are reported, for the sake of completeness, in Appendix B.

In a recent work Brock et al. (2020) argued that initial weights should not be considered as random variables, but are rather the realization of a random process. Thus, weights mean and variance are empirical values different from those of the generating random process. Hence, normalization of the weights matrix should be performed after sampling to obtain the desired moments. Moreover, they argue that channel-wise responses should be analyzed. This leads to the different initialization strategy:

$$\text{Var}[W_i] = \frac{2/(1 - \frac{1}{\pi})}{n_{\text{in}}}, \tag{3}$$

where W_i is a single channel of the filter. Note that if mean and variance are preserved channel-wise, then they are also preserved if the whole layer is taken into account.

The authors do not take into account the *backward variance*. Brock et al. (2020) show that the latter initialization scheme allows to experimentally preserve the channel-wise activation variance, whereas He's technique only works at the full-layer level.

In the ResNet setting, initialization alone is not sufficient to make the training properly work without batch normalization, if the commonly employed architecture with Identity Shortcuts (see Figure 1a) is considered.

In particular, the skip-branch summation

$$x_l = x_{l-1} + f_l(x_{l-1}), \tag{4}$$

at the end of each block does not preserve variance, causing the phenomenon known as *internal covariate shift* (Ioffe & Szegedy, 2015).

In order to overcome this issue, Batch Normalization has been devised. More recently, effort has been put into designing other architectural and algorithmic modifications that dot not rely on batch statistics.

Specifically, Zhang et al. (2019); De & Smith (2020); Bachlechner et al. (2020) modified the skip-identity summation as to downscale the variance at the beginning of training, biasing, in other words, the network towards the identity function, i.e., computing

$$x_{l+1} = x_{l-1} + \alpha f_l(x_{l-1}).$$

This has the downside that α must be tuned and is dependent on the number of layers. Moreover, while these solutions enjoy good convergence on the training set, they appear not to be sufficient to make deep ResNets reach state-of-the-art test accuracies (Brock et al., 2020).

More recently, Brock et al. (2020) proposed to additionally perform a runtime layer-wise normalization of the weights, together with the empirical channel-wise intialization scheme. However, we show in the following that the latter scheme, while enjoying perfectly conserved forward variances, induces the network to work in a regime of *exploding gradients*, i.e., the variance of the gradients of the shallowest layers is exponentially larger than that of the deepest ones. Reasonably, Brock et al. (2021) found the use of a tailored adaptive gradient clipping to be beneficial because of this reason.

3 The Proposed Method

In order to overcome the issue discussed at the end of the previous section, we propose to modify the summation operation of ResNet architectures so that, at the beginning of the training, the mean of either the activations or the gradients is zero and the variance is preserved throughout the network. In our view, our proposal is a natural extension of the work of He et al. (2015) for the case of ResNet architectures. Note that, to develop an effective initialization scheme, the residual block summation has to be slightly modified.

Namely, we propose the following general scheme (see Figure 1b):

$$x_l = c \cdot \left(h(x_{l-1}) + f_l(x_{l-1})\right), \tag{5}$$

where c is a suitable constant, h is a generic function operating on the skip branch and $f_l(x_{l-1})$ represents the output of the convolutional branch. We assume that we are able, through a proper initialization, to have zero mean and controlled variance (either backward or forward) for each block f_l.

In a typical ResNet architecture, f_l is a sequence of two or three convolutions, each one preceded by a ReLU activation - *pre-activation* (He et al., 2016b) - allowing to control both mean and variance through initialization schemes (1) and (2). Note that *post-activated* ResNets do not allow f_l to have zero (either gradient or activation) mean, which corroborates the analysis done by He et al. (2016a).

We perform the analysis in this general setting, deriving the condition h and c must satisfy in order to preserve either the forward or backward variance. Then, we propose different ways in which h and c can be defined to satisfy such conditions.

3.1 The Forward Case

Firstly, we note that initializing the weights of each block f following rule (1), hypothesizing that $\mathbb{E}[x_{l-1}] = 0$ and $\text{Var}[x_{l-1}] = 1$, we can easily obtain that

$$\mathbb{E}[f_l(x_{l-1})] = \mathbb{E}[x_{l-1}] = 0, \qquad \text{Var}[f_l(x_{l-1})] = \text{Var}[x_{l-1}] = 1.$$

We can also make the reasonable assumption that $f_l(x_{l-1})$ and $h(x_{l-1})$ have zero correlation, thus, getting

$$\mathbb{E}[x_l] = c \cdot (\mathbb{E}[h(x_{l-1})]) + \mathbb{E}[f_l(x_{l-1})]) = c \cdot \mathbb{E}[h(x_{l-1})]$$
$$\text{Var}[x_l] = c^2 \cdot (\text{Var}[h(x_{l-1})] + \text{Var}[f_l(x_{l-1})]) = c^2 \cdot (\text{Var}[h(x_{l-1})] + 1),$$

i.e., the activation signal can be preserved by defining h so that $\mathbb{E}[h(x_{l-1})] = 0$ and $\text{Var}[h(x_{l-1})] = \frac{1}{c^2} - 1$.

3.2 The Backward Case

For the gradients at layer $l-1$, we have

$$\frac{\partial \mathcal{L}}{\partial x_{l-1}} = \frac{\partial \mathcal{L}}{\partial x_l}\frac{\partial x_l}{\partial x_{l-1}} = c \cdot \frac{\partial \mathcal{L}}{\partial x_l}\left(\frac{\partial h(x_{l-1})}{\partial x_{l-1}} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right).$$

We can assume by induction that the gradients at layer l have zero mean, i.e., $\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] = 0$. Then, we get

$$\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = c \cdot \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\frac{\partial x_l}{\partial x_{l-1}}\right] = c \cdot \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\mathbb{E}\left[\frac{\partial x_l}{\partial x_{l-1}}\right] = 0.$$

Assuming zero correlation between $\frac{\partial \mathcal{L}}{\partial x_l}$ and $\frac{\partial x_l}{\partial x_{l-1}}$, we can further write

$$\begin{aligned}
\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = c^2 \cdot \Bigg(&\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\text{Var}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right] \\
&+ \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\mathbb{E}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right]^2 \\
&+ \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]^2 \text{Var}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right]\Bigg) \\
= c^2 \cdot \Bigg(&\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\left(\text{Var}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}}\right] + \text{Var}\left[\frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right]\right) \\
&+ \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\left(\mathbb{E}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}}\right] + \mathbb{E}\left[\frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right]\right)^2 \\
&+ \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]^2 \left(\text{Var}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}}\right] + \text{Var}\left[\frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right]\right)\Bigg).
\end{aligned}$$

Now, we also know that, if we initialize the weight of each block f_l by rule (2), it holds $\mathbb{E}\left[\frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right] = 0$ and $\text{Var}\left[\frac{\partial f_l(x_{l-1})}{\partial x_{l-1}}\right] = 1$. Therefore we can conclude

$$\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = c^2 \cdot \left(\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\left(\text{Var}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}}\right] + 1\right) + \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\mathbb{E}\left[\frac{\partial h(x_{l-1})}{\partial x_{l-1}}\right]^2\right). \quad (6)$$

The preservation of the gradients signal can thus be obtained by suitably defined h and c.

We argue that some of the techniques proposed by Brock et al. (2020; 2021) to train deep Residual Networks (weight normalization layers, adaptive gradient clipping, etc.) become necessary because initialization (3) focuses on the preservation of the forward activation signal while disregarding the backward one.

Indeed, the correction factor γ_g^2 in equation 3 breaks the conservation property of the gradients signal, as opposed to equation 1. As we back-propagate through the model, the factor γ_g^2 amplifies the gradients signal at each layer, so that the gradients at the last layers are orders of magnitude larger than those at the first layers (going from output to input layers), i.e., the network is in a regime of *exploding gradient*. In the section devoted to the numerical experiments we will show the forward and backward behaviour of these nets.

3.3 GRADIENTS SIGNAL PRESERVING SETUPS

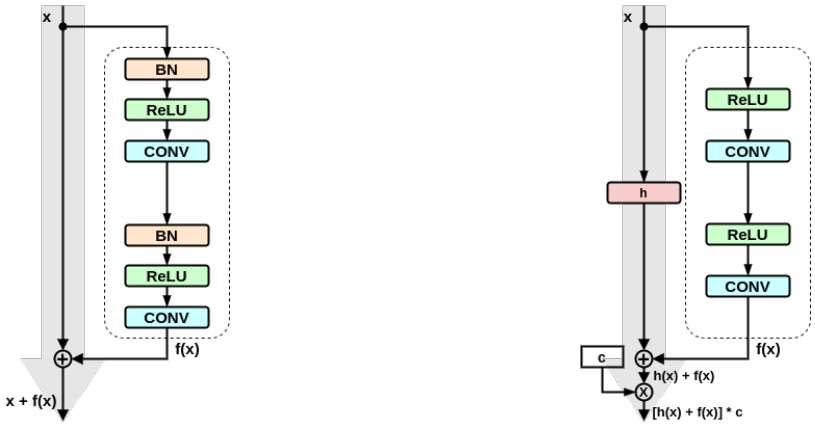
It is well know that *exploding gradients* make training hard (from an optimization perspective). Indeed, without further algorithmic or architectural tricks we are unable to train very deep networks. It is important to note that in the seminal analyses from Glorot & Bengio (2010) and He et al. (2015) the derivation implied that preserving the forward variance entailed preserving also the backward variance too (at least to some reasonable amount). Indeed forward and backward variance can be equally preserved if, as already noted, for each layer, the number of input and output channels is equal. On the contrary, in the derivation of Brock et al. (2020; 2021), this relationship between forward and backward variance is lost so that conserving the forward variance implies *exploding gradients*.

For this reason, in the following we mainly focus on the backwards signal, which we argue being a more important thing to look at when forward and backward variance are not tightly related. For this reason, we propose three different possible schemes for choosing c and h in equation 5. In particular:

1. **scaled identity shortcut (IdShort):** $h(x) = x$, $c = \sqrt{0.5}$.
   This choice, substituting in equation 6, leads to
   $$\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = 0.5 \cdot \left(\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right](0+1) + \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] \cdot 1\right) = \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right],$$
   i.e., the variance of gradients is preserved. As for the activations, we get $\mathbb{E}[x_l] = 0$ and $\text{Var}[x_l] = 0.5 \cdot (1+1) = 1$, i.e., activations signal preservation, for all layers where input and output have the same size.
   Note that the latter scheme is significantly different from approaches, like those from Zhang et al. (2019); De & Smith (2020); Bachlechner et al. (2020), that propose to add a (learnable) scalar that multiplies the skip branch. In fact, in the proposed scheme the (constant) scalar multiplies both branches and aims at controlling the total variance, without biasing the network towards the identity like in the other approaches.
   This is the simplest variance preserving modification of the original scheme that can be devised, only adding a constant scalar scaling at the residual block.
2. **scaled identity shortcut with a learnable scalar (LearnScalar):** $h(x) = \alpha x$, α initialized at 1, $c = \sqrt{0.5}$. In equation 6 we again get at initialization
   $$\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = 0.5 \cdot \left(\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right](0+1) + \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] \cdot \alpha^2\right) = \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right],$$
   and similarly as above we also obtain the forward preservation at all layers with $N = \hat{N}$.
3. **scaled identity shortcut with a (1×1)-strided convolution (ConvShort):** $h(x) = W_s x$ initialized by (2), $c = \sqrt{0.5}$. Since we use He initialization on the convolutional shortcut (He et al., 2016b), we have $\mathbb{E}\left[\frac{\partial h((x_{l-1})}{\partial x_{l-1}}\right] = 0$ and $\text{Var}\left[\frac{\partial h((x_{l-1})}{\partial x_{l-1}}\right] = 1$, hence we obtain in equation 6
   $$\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = 0.5 \cdot \left(\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right](1+1) + \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right]\right) = \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right].$$
   Again, if we consider the layers with equal size for inputs and outputs, we also get $\mathbb{E}[x_l] = 0$ and $\text{Var}[x_l] = 0.5 \cdot (1+1) = 1$.
   Note that this setting (without the scale factor) is commonly used in most ResNet architectures when x_{l-1} and $f_l(x_{l-1})$ have not the same pixel resolution (for instance because f contains some strided convolution) or the same number of channels.

4 EXPERIMENTS

We start the investigation by numerically computing forward and backward variances for the different initialization schemes. We employ the recently introduced Signal Propagation Plots (Brock et al., 2020) for the forwards variance and a modification that looks at the gradients instead of the activations for the backwards case.



(a) Standard pre-activated Residual Block

(b) Generalized Normalizer-Free Residual Block

Figure 1: Architectures of Residual Blocks. For both pictures the grey arrow marks the easiest path to propagate the information.

We employ the ResNet-50 architecture to extract the plots. Other architectures are reported in Appendix A. Notably, the pattern emerged for ResNet-50 is clearly visible also for the other ResNet architectures.

In particular we extract the plots for

- classical ResNet with He initialization, *fan in* mode (1) and *fan out* mode (2);
- same of the preceding with batch normalization;
- ResNet with the three proposed residual summation modifications and their proper intialization to preserve the backwards variance[1];
- same as the preceding but employing the intialization of Brock et al. (2020).

For all the initialization schemes, we perform the empirical standardization to zero mean and desired variance of weights at each layer, after the random sampling.

From Figure 2 we first note that, as already pointed out by Brock et al. (2020), classical ResNets with He initialization do not preserve neither forwards nor backwards signals while the use of batch normalization manages to fix things up.

Next, we note that employing the proposed strategies (with proper initialization) we are able to conserve the variance of the gradients. On the contrary, the initialization proposed by Brock et al. (2020) amazingly preserves the forward signal but puts the network in a regime of exploding gradients. Namely, the variance of the gradients exponentially increases going from the deepest to the shallowest residual layers. Additionally, we can also note how the proposed strategies also preserve the activations variance, up to some amount, while when employing the scheme of Brock et al. (2020) the relationship between forward and backward variance is lost.

We continue the analysis by performing a set of experiments on the well-known CIFAR-10 dataset (Krizhevsky et al., 2009) in order to understand if an effective training can be actually carried out under the different schemes and compare them in terms of both train and test accuracy. In particular, we are interested in checking out if the proposed schemes can reach batch normalization test performance.

All the experiments described in what follows have been performed using SGD with an initial learning rate of 0.01, a momentum of 0.9 and a batch size of 128, in combination with a Cosine Annealing scheduler (Loshchilov & Hutter, 2016) that decreases the learning rate after every epoch. Moreover, in

[1]Note that, as in the standard implementation, in IdShort and LearnScalar we employ ConvShort when x has not the same pixel resolution or number of channels of $f(x)$.

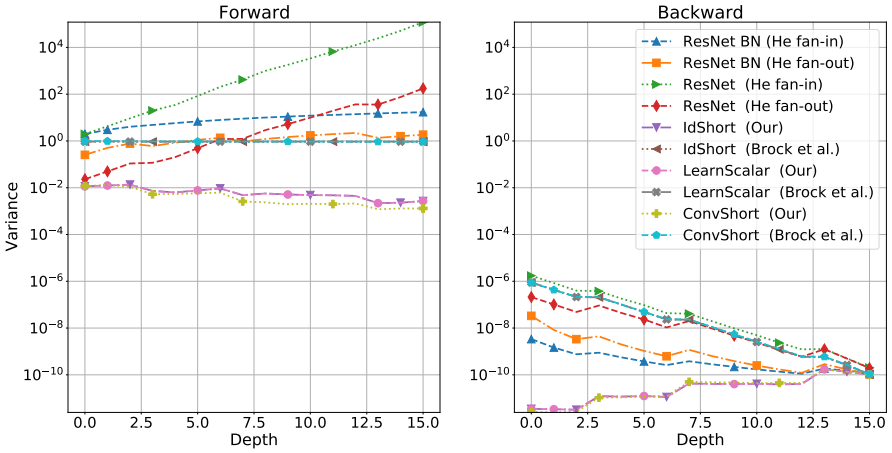


Figure 2: Signal propagation plots representing the variance of the forward activations (on the left) and the backward gradients variance (on the right) under our proposed initialization scheme: both values refer to residual block output. Both signals depict the behaviour for a ResNet-50 with convolutional shortcuts. The x-axis is the residual layer depth, while on the y-axis the variance of the signal is reported in a logarithmic scale.

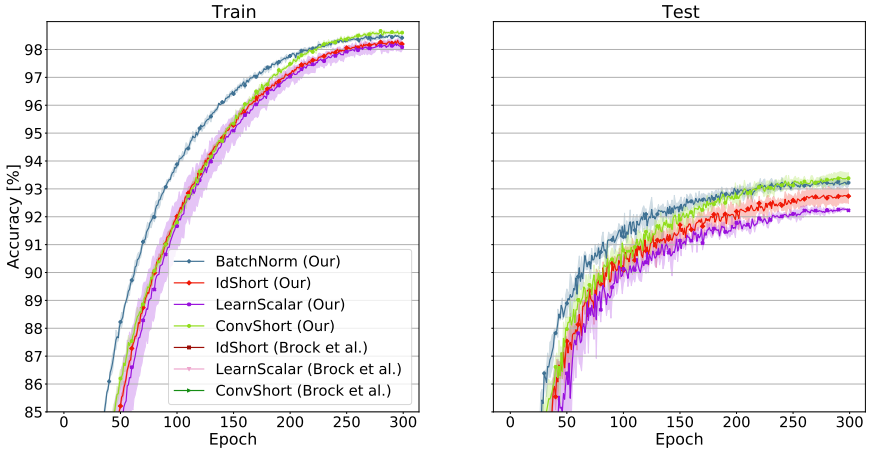


Figure 3: Test and Train accuracies of ResNet-50 under different combinations of residual block modifications and initialization: standard ResNet with BatchNorm and IdShort, LearnScalar, ConvShort using both Brock et al. (2020) and ours initialization. Each experiment has been run three times: the solid line is the mean value while the surrounding shadowed area represents the standard deviation. Finally, the x-axis is the epoch at the which the accuracy (reported in the y-axis) has been computed.

addition to the standard data augmentation techniques, we have also employed the recently proposed RandAugment method (Cubuk et al., 2020).

In Figure 3 both train and test accuracies are shown for all the configurations. The results report the mean and the standard deviation of three independent runs.

The first thing to notice is that with the initialization scheme of Brock et al. (2020) we are unable to train the network (the curve is actually absent from the plot). This is due to the fact that the network, at the start of the training, is in a regime of exploding gradients, as observed in the SPPs.

On the contrary, we can see how, thanks to the correct preservation of the backward signals, training is possible for all the proposed schemes when a gradient preserving initialization scheme is employed.

Finally, we notice that, while all the schemes achieve satisfactory test accuracies, only the *ConvShort* modification has an expressive power able to close the gap (and even outperform at the last epochs) with the network trained using with Batch Normalization. Thus, according to Figure 3 (and also to the additional experiments in Appendix A), *ConvShort* appears to be an architectural change that, in combination with the proposed initialization strategy, is able to close the gap with a standard pre-activated ResNet with Batch Normalization.

5 Conclusion

In this work we proposed a slight architectural modification of ResNet-like architectures that, coupled with a proper weights initialization, can train deep networks without the aid of Batch Normalization. Such initialization scheme is general and can be applied to a wide range of architectures with different building blocks. Importantly, our strategy does not require any additional regularization nor algorithmic modifications, as compared to other approaches. We show that this setting achieves competitive results on CIFAR-10.

Author Contributions

This Section is hidden for double-blind review

Acknowledgements

This Section is hidden for double-blind review

Ethics Statement

Authors have no ethical concern to report.

References

Devansh Arpit, Yingbo Zhou, Bhargava Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *International Conference on Machine Learning*, pp. 1168–1176. PMLR, 2016.

Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/e520f70ac3930490458892665cda6620-Paper.pdf.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian J. McAuley. Rezero is all you need: Fast convergence at large depth. *CoRR*, abs/2003.04887, 2020. URL https://arxiv.org/abs/2003.04887.

Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *International Conference on Learning Representations*, 2020.

Andrew Brock, Soham De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. *ArXiv*, abs/2102.06171, 2021.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

Yann N Dauphin and Samuel Schoenholz. Metainit: Initializing learning by learning to initialize. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/876e8108f87eb61877c6263228b67256-Paper.pdf.

Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33, 2020.

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. JMLR Workshop and Conference Proceedings, 2010.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d81f9c1be2e08964bf9f24b15f0e4900-Paper.pdf.

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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1729–1739, Red Hook, NY, USA, 2017. Curran Associates Inc.

Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2822–2830. IEEE Computer Society, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Janghyeon Lee, Donggyu Joo, Hyeong Gwon Hong, and Junmo Kim. Residual continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4553–4560, 2020.

Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, pp. 989–998, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization, 2019.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization, 2020.

Nicholas D Socci, Daniel D Lee, and H Sebastian Seung. The rectified gaussian distribution. *Advances in Neural Information Processing Systems*, pp. 350–356, 1998.

Cecilia Summers and Michael J. Dinneen. Four things everyone should know to improve batch normalization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJx8HANFDH.

Junjie Yan, Ruosi Wan, Xiangyu Zhang, Wei Zhang, Yichen Wei, and Jian Sun. Towards stabilizing batch statistics in backward propagation of batch normalization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgGjRVKDS.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Residual learning without normalization via better initialization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gsz30cKX.

A ADDITIONAL EXPERIMENTS

In the present Appendix we report additional experiments on other architectures, namely ResNet18 and ResNet-101, that were excluded from the main body of the paper (due to maximum length constraint). We report both the Signal Propagation Plots (Brock et al., 2020) and the train and test accuracy for CIFAR-10.

From the SPPs, shown in in Figures 4 and 5, we can observe that the same patterns that emerged for ResNet-50 are also present in the other architectures. Interestingly, we note that the observed trends are more conspicuous in deeper networks.

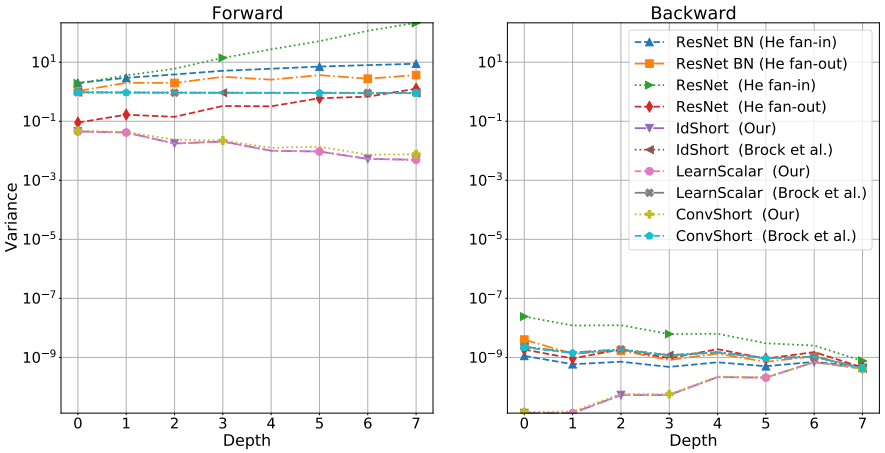


Figure 4: Signal propagation plots representing the variance of the forward activations (on the left) and the backward gradients variance (on the right) under our proposed initialization scheme: both values refer to residual block output. Both signals depict the behaviour for a ResNet-18 with convolutional shortcuts. The x-axis is the residual layer depth, while on the y-axis the variance of the signal is reported in a logarithmic scale.

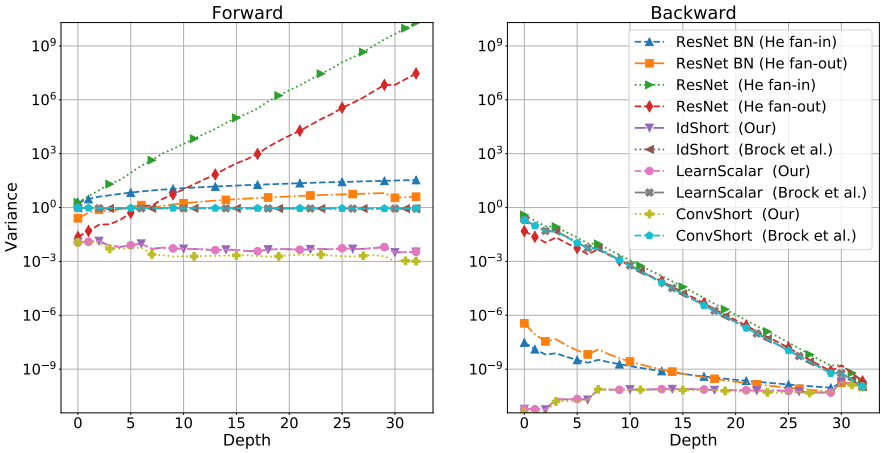


Figure 5: Signal propagation plots representing the variance of the forward activations (on the left) and the backward gradients variance (on the right) under our proposed initialization scheme: both values refer to residual block output. Both signals depict the behaviour for a ResNet-101 with convolutional shortcuts. The x-axis is the residual layer depth, while on the y-axis the variance of the signal is reported in a logarithmic scale.

Similarly, from CIFAR-10 accuracies, reported in Figures 6 and 7, we can see that the same findings obtained for ResNet-50 carry over to the other architectures. While we were able to train all the network configurations on ResNet-18 we were unable to do so for the deeper ResNet-101 when non properly initialized. Moreover, once again, we found *ConvShort* to be the best configuration in ResNet-101 although only the second-best in ResNet-18.

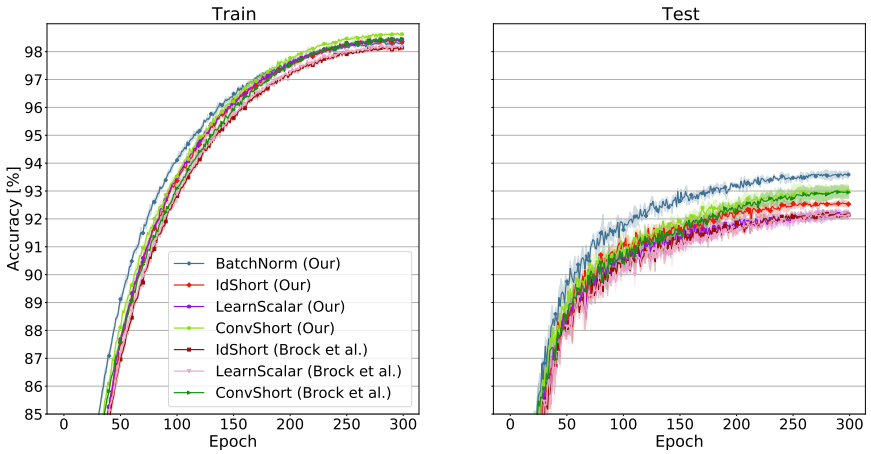


Figure 6: Test and Train accuracies of ResNet-18 under different combinations of residual block modifications and initialization: standard ResNet with BatchNorm and IdShort, LearnScalar, ConvShort using both Brock et al. and ours initialization. Each experiment has been run three times: the solid line is the mean value while the surrounding shadowed area represents the standard deviation. Finally, the x-axis is the epoch at the which the accuracy (reported in the y-axis) has been computed.

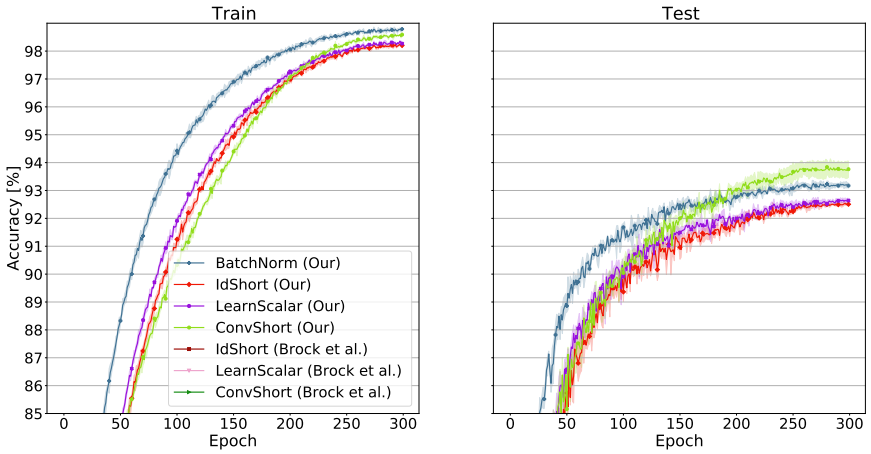


Figure 7: Test and Train accuracies of ResNet-101 under different combinations of residual block modifications and initialization: standard ResNet with BatchNorm and IdShort, LearnScalar, ConvShort using both Brock et al. and ours initialization. Each experiment has been run three times: the solid line is the mean value while the surrounding shadowed area represents the standard deviation. Finally, the x-axis is the epoch at the which the accuracy (reported in the y-axis) has been computed.

Finally we also report the number of parameters and FLOPs for the considered architecture in Table 1. We note that *ConvShort* and *BatchNorm* have the same computational cost while the others configurations can be employed as more light-weight alternatives.

Model	Input Resolution	Params (M)	#FLOPs (G)
ResNet-18 BatchNorm	$32 \times 32 \times 3$	11.52	1.16
ResNet-18 IdShort	$32 \times 32 \times 3$	11.16	1.12
ResNet-18 LearnScalar	$32 \times 32 \times 3$	11.16	1.12
ResNet-18 ConvShort	$32 \times 32 \times 3$	11.52	1.16
ResNet-50 BatchNorm	$32 \times 32 \times 3$	38.02	4.2
ResNet-50 IdShort	$32 \times 32 \times 3$	23.47	2.6
ResNet-50 LearnScalar	$32 \times 32 \times 3$	23.47	2.6
ResNet-50 ConvShort	$32 \times 32 \times 3$	38.02	4.2
ResNet-101 BatchNorm	$32 \times 32 \times 3$	74.78	8.92
ResNet-101 IdShort	$32 \times 32 \times 3$	42.41	5.02
ResNet-101 LearnScalar	$32 \times 32 \times 3$	42.41	5.02
ResNet-101 ConvShort	$32 \times 32 \times 3$	74.78	8.92
ResNet-152 BatchNorm	$32 \times 32 \times 3$	105.06	13.62
ResNet-152 IdShort	$32 \times 32 \times 3$	58.01	3.72
ResNet-152 LearnScalar	$32 \times 32 \times 3$	58.01	3.72
ResNet-152 ConvShort	$32 \times 32 \times 3$	105.06	13.62

Table 1: Computational cost and number of parameters of the considered architectures.

B Deriving Standard Initialization Schemes

B.1 He Initialization

Consider the response of each layer l

$$z_l = g(x_{l-1}), \qquad x_l = W_l z_l,$$

where x is a k^2c-by-1 vector that represents co-located $k \times k$ pixels in c input channels, W_l is a d-by-n matrix where d is the number of filters and $g(\cdot)$ is a nonlinear activation function. In the following, we will consider the classical ReLU, $g(z) = \max(0, z)$.

Formally, let us consider normally distributed input data $x_{l-1} \sim \mathcal{N}(0, \sigma^2)$. It is well known that with this particular activation we get a Rectified Normal Distribution (Socci et al., 1998; Arpit et al., 2016) with central moments:

$$\mu_g = \mathbb{E}[g(z_l)] = \frac{\sigma}{\sqrt{2\pi}}, \qquad \sigma_g^2 = \text{Var}[g(z_l)] = \frac{\sigma^2}{2} - \frac{\sigma^2}{2\pi}.$$

From the basic properties of variance we also have

$$\mathbb{E}[g(z_l)^2] = \mu_g^2 + \sigma_g^2 = \frac{\sigma^2}{2}. \tag{7}$$

Making the assumption that the weights W (we omit for simplicity the dependency on layer l) at a network's layer l are i.i.d. with zero mean ($\mu_W = 0$) and that have zero correlation with the input (hence $\text{Var}[Wg(x_{l-1})] = \sigma_W^2 \sigma_g^2$, putting $Var[W] = \sigma_W^2$), we obtain for the output elements that

$$\mathbb{E}[x_l] = n_{\text{in}} \mu_g \mu_W = 0$$

and

$$\begin{aligned} \text{Var}[x_l] &= n_{\text{in}}[\mathbb{E}[W^2 z_l^2] - (\mathbb{E}[W z_l])^2] = n_{\text{in}}[\mathbb{E}[W^2]\mathbb{E}[z_l^2] - (\mathbb{E}[W]\mathbb{E}[z_l])^2] \\ &= n_{\text{in}}[(\mu_W^2 + \sigma_W^2)(\mu_g^2 + \sigma_g^2) - \mu_W^2 \mu_g^2] = n_{\text{in}}[\sigma_g^2(\mu_W^2 + \sigma_W^2) + \sigma_W^2 \mu_g^2] \\ &= n_{\text{in}}[\sigma_W^2(\mu_g^2 + \sigma_g^2)]. \end{aligned} \tag{8}$$

where $n_{\text{in}} = k^2 c$ with k the filter dimension and c the number of input channels (*fan in*). Hence if input has unit variance ($\sigma^2 = 1$) we obtain output unit variance by initializing W in such a way that

$$\text{Var}[W] = \frac{2}{n_{\text{in}}}. \tag{9}$$

Similarly we can perform the analysis w.r.t. the gradients signal.

For back-propagation, we can also write

$$\frac{\partial \mathcal{L}}{\partial z_l} = \hat{W} \frac{\partial \mathcal{L}}{\partial x_l},$$

where $\mathcal{L}$ is the loss function and $\hat{W}$ is a suitable rearrangement of W. If weights W are initialized with zero mean from a symmetric distribution, $\frac{\partial \mathcal{L}}{\partial z_l}$ will also have zero mean. We can assume $\frac{\partial \mathcal{L}}{\partial x_l}$ and $\hat{W}$ to be uncorrelated.

In addition,

$$\frac{\partial \mathcal{L}}{\partial x_{l-1}} = \frac{\partial \mathcal{L}}{\partial x_l} g'(x_{l-1});$$

being g the ReLU, $g'(x_{l-1})$ is either 0 or 1 with equal probability, hence, assuming $g'(x_{l-1})$ and $\frac{\partial \mathcal{L}}{\partial x_l}$ uncorrelated, we get

$$\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = \frac{1}{2}\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] = \frac{1}{2}\mathbb{E}\left[\hat{W}\right]\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = 0,$$
$$\mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial x_{l-l}}\right)^2\right] = \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-l}}\right] = \frac{1}{2}\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right].$$

Therefore, we can conclude that

$$\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = \frac{1}{2}\text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] = \frac{1}{2}\text{Var}\left[\hat{W}\frac{\partial \mathcal{L}}{\partial x_l}\right] = \frac{n_{\text{out}}}{2}\sigma_W^2 \text{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right].$$

Thus, the initialization

$$\text{Var}[W] = \frac{2}{n_{\text{out}}}, \tag{10}$$

where $n_{\text{out}} = k^2 d$ with k the filter dimension and d the number of output channels (*fan out*), allows to preserve the variance of gradients.

B.2 BROCK INITIALIZATION

In contrast with the analysis from Appendix B.1, here initial weights are not considered as random variables, but are rather the realization of a random process. Thus, weights mean and variance are empirical values different from those of the generating random process. In order to satisfy the suitable assumptions of the analysis, weights should be actually re-normalized to have empirical zero mean and predefined variance.

Moreover, the channel-wise responses are analyzed. The derivations in equation 8 should be revised in order to consider expected value and the variance of any single channel i of the output x_l and to take into account constant $\sigma_{W_i}^2$ and $\mu_{W_i} = 0$; specifically, we obtain

$$\text{Var}[x_{l\,i}] = \sum_{j=1}^{n_{\text{in}}} \text{Var}[W_{ij} z_{l\,j}] = \sum_{j=1}^{n_{\text{in}}} W_{ij}^2 \text{Var}[z_{l\,j}]$$
$$= n_{\text{in}} \left(\sigma_g^2 \cdot \frac{1}{n_{\text{in}}} \sum_{j=1}^{n_{\text{in}}} W_{ij}^2\right) = N\sigma_g^2(\mu_{W_i}^2 + \sigma_{W_i}^2) = n_{\text{in}}\sigma_g^2\sigma_{W_i}^2,$$

so that we retrieve the following initialization rule to preserve an activation signal with unit variance:

$$\text{Var}[W_i] = \frac{\gamma_g^2}{n_{in}}, \tag{11}$$

where $\gamma_g^2 = \frac{2}{1-\frac{1}{\pi}}$ for the ReLU activation. Note that if mean and variance are preserved channel-wise, then they are also preserved if the whole layer is taken into account.