A Robust Initialization of Residual Blocks for Effective ResNet Training Without Batch Normalization

Enrico Civitelli, Alessio Sortino, Matteo Lapucci, Francesco Bagattini, and Giulio Galvan

***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 brief, 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, CIFAR-100 and ImageNet without further regularization nor algorithmic modifications.**

***Index Terms*— Exploding gradient, normalization-free ResNets, residual blocks, weights initialization.**

I. Introduction

Batch normalization [18], in conjunction with skip connections [12], [14], [15], has allowed the training of significantly deeper and more expressive 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 [16], [23].

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 in [4], [5], and [27] 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 [28] and breaks the independence between training examples in a minibatch, which can be extremely harmful for some learning tasks [20], [21].

For these reasons, a new stream of research emerged which aims at removing batch normalization from modern architectures. Several works [3], [8], [30] 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.

Manuscript received 17 October 2022; revised 19 July 2023 and 1 August 2023; accepted 15 October 2023. *(Corresponding author: Matteo Lapucci.)*

Enrico Civitelli, Alessio Sortino, and Matteo Lapucci are with the Dipartimento di Ingegneria dell'Informazione, Università di Firenze, 50139 Florence, Italy (e-mail: matteo.lapucci@unifi.it).

Francesco Bagattini and Giulio Galvan are with Flair-Tech, 50135 Florence, Italy.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2023.3325541.

Digital Object Identifier 10.1109/TNNLS.2023.3325541

In [26], authors propose a different modification of the standard residual layer, suitably carrying out a weighted sum of the identity and the nonlinear branches.

More recently [4], [5], an approach was proposed that combines a modification of the residual block with a careful initialization, a variation of the Scaled Weight Standardization [17], [25] 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.

1) We show that while *NFNets* from [4], [5] enjoy a perfect forward variance (as already noted in [4]), it puts the network in a regime of *exploding gradients*. This is shown by looking at the variance of the derivatives of the loss with respect to to the feature maps at different depths.
2) We propose a simple modification of the residual layer and then develop a suitable initialization scheme building on the work of [13].
3) We show that the proposed architecture achieves competitive results on CIFAR-10, CIFAR-100, and ImageNet [19], which we consider evidence supporting our theoretical claims.

II. Background

As highlighted in a number of recent studies [1], [7], [11], 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 in [10] and later extended in [13]. 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 [13] investigates the variance of each response layer l (*forward variance*)

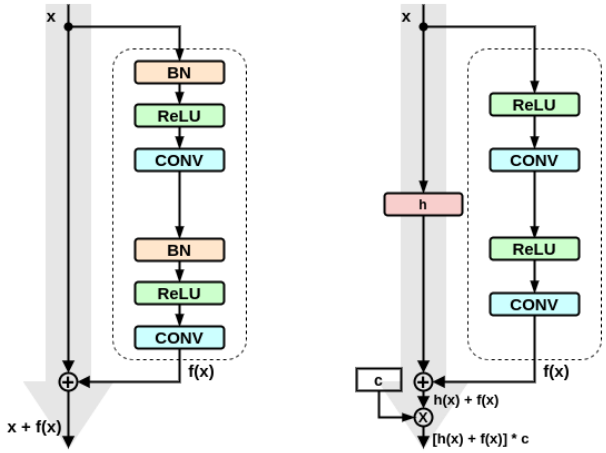
$$z_l = \mathrm{ReLU}(x_{l-1}), \qquad x_l = W_l z_l.$$

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

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

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 with respect to each layer response (*backward variance*) $\partial \mathcal{L}/\partial x_l$.



(a) (b)

Fig. 1. Architectures of Residual Blocks. For both pictures, the gray arrow marks the easiest path to propagate the information. (a) Standard preactivated residual block. (b) generalized normalizer-free residual block.

In this case, we can preserve zero mean and constant variance if we have

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

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

Note that (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 in [13] should generally lead to the conservation of both *forward* and *backward* signals.

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

In a recent work [4], it is argued that initial weights should not be considered as random variables, but are rather the realization of a random process. Thus, empirical mean and variance of the weights should do not coincide with the moments 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

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

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*. [4] 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 Fig. 1(a)] 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* [18].

In order to overcome this issue [2], 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, in [3], [8], [30], the skip-identity summation was modified as to downscale the variance at the beginning of training, biasing, in other words, the network toward 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 [4].

Similarly, the authors of [26] suggest to compute the output of the residual branch as a weighted sum between the identity and the nonlinear branch. Formally, the residual layer becomes

$$x_l = \alpha_l x_{l-1} + \beta_l f(x_{l-1})$$

where coefficients α_l and β_l can be set so that the *forward variance* is conserved by imposing that $\alpha_l^2 + \beta_l^2 = 1$. Different strategies can be employed to choose their relative value.

More recently, [4] proposed to additionally perform a runtime layer-wise normalization of the weights, together with the empirical channel-wise initialization 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, the use of a tailored adaptive gradient clipping was found to be beneficial because of this reason [5].

III. The Proposed Method

In order to overcome the issue discussed at the end of Section II, 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 in [13] 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 analyze the following general scheme [see Fig. 1(b)]:

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

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. As we will detail later in this work, this framework generalizes the most commonly employed skip connections.

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* [15] - allowing to control both mean and variance through initialization schemes (1) and (2). Note that *postactivated* ResNets do not allow f_l to have zero (either gradient or activation) mean, which corroborates the analysis done in [14].

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.

A. Forward Case

Let us assume that $\mathbb{E}[x_0] = 0$ and $\mathrm{Var}[x_0] = 1$, being x_0 the input data, and let us reason by induction.

By the inductive step, we assume $\mathbb{E}[x_{l-1}] = 0$ and $\mathrm{Var}[x_{l-1}] = 1$; if weights of each block f are initialized following rule (1), we can easily verify that

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

Recalling [26], we are allowed to assume that $f_l(x_{l-1})$ and $h(x_{l-1})$ have zero correlation, thus, getting

$$\begin{aligned}\mathbb{E}[x_l] &= c \cdot \left(\mathbb{E}\left[h(x_{l-1})\right]\right) + \mathbb{E}\left[f_l(x_{l-1})\right]\right) \\ &= c \cdot \mathbb{E}\left[h(x_{l-1})\right], \\ \mathrm{Var}[x_l] &= c^2 \cdot \left(\mathrm{Var}\left[h(x_{l-1})\right] + \mathrm{Var}\left[f_l(x_{l-1})\right]\right) \\ &= c^2 \cdot \left(\mathrm{Var}\left[h(x_{l-1})\right] + 1\right).\end{aligned}$$

Thus, defining h so that $\mathbb{E}[h(x_{l-1})] = 0$ and $\mathrm{Var}[h(x_{l-1})] = (1/c^2) - 1$, the activation signal can be preserved and the induction step established.

B. Backward Case

Let us assume that for the gradients at the output layer L, we have $\mathbb{E}[(\partial\mathcal{L}/\partial x_L)] = 0$ and $\mathrm{Var}[(\partial\mathcal{L}/\partial x_L)] = C$ and that we initialize the weight of each block f_l by rule (2).

Now, we can assume by induction that the gradients at layer l have zero mean and preserved variance, i.e., $\mathbb{E}[(\partial\mathcal{L}/\partial x_l)] = 0$ and $\mathrm{Var}[(\partial\mathcal{L}/\partial x_l)] = C$. Since 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 get

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

Moreover, under the reasonable assumption that there is zero correlation between $(\partial\mathcal{L}/\partial x_l)$ and $(\partial x_l/\partial x_{l-1})$, we can further write

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

Thanks to the initialization rule (2), it holds $\mathbb{E}[(\partial f_l(x_{l-1})/\partial x_{l-1})] = 0$ and $\mathrm{Var}[(\partial f_l(x_{l-1})/\partial x_{l-1})] = 1$. Therefore, we can conclude

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

The induction step can therefore be established and the preservation of the gradients signal obtained by suitably defined h and c.

We argue that some of the techniques proposed by [4], [5] 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 $\gamma_g^2 = 2/(1-(1/\pi))$ in (3) breaks the conservation property of the gradients signal, as opposed to (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 behavior of these nets.

C. Gradients Signal Preserving Setups

It is well known 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 [10] and [13], 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 in [4], [5], 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 (5).

1) *Scaled Identity Shortcut (IdShort):* $h(x) = x$, $c = (0.5)^{1/2}$. This choice, substituting in (6), leads to

$$\begin{aligned}\mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_{l-1}}\right] &= \frac{1}{2} \cdot \left(\mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right] \cdot 1 + \mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right] \cdot 1\right) \\ &= \mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right]\end{aligned}$$

i.e., the variance of gradients is preserved. As for the activations, we get $\mathbb{E}[x_l] = 0$ and

$$\mathrm{Var}[x_l] = 0.5 \cdot (1+1) = 1$$

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 [3], [8], [30], 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 toward 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 (Learn-Scalar):* $h(x) = \alpha x$, α initialized at 1, $c = (0.5)^{1/2}$. In (6), we again get at initialization

$$\begin{aligned}\mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_{l-1}}\right] &= \frac{1}{2} \cdot \left(\mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right] \cdot 1 + \mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right] \cdot \alpha^2\right) \\ &= \mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial x_l}\right]\end{aligned}$$

and similarly as above, we also obtain the forward preservation at all layers with $N = \hat{N}$.

3) *Scaled Identity Shortcut With a* **(1 x 1)***-Strided Convolution (ConvShort):* $h(x) = W_s x$ initialized by (2), $c = (0.5)^{1/2}$. Since we use He initialization on the convolutional shortcut [15], we have $\mathbb{E}[(\partial h(x_{l-1})/\partial x_{l-1})] = 0$ and $\mathrm{Var}[\partial h(x_{l-1})/\partial x_{l-1}] = 1$, hence, we obtain in (6)

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{l-1}}\right] = \frac{1}{2} \cdot \left(\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] \cdot 2 + \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_l}\right] \cdot 0\right).$$

Again, if we consider the layers with equal size for inputs and outputs, we also get $\mathbb{E}[x_l] = 0$ and $\mathrm{Var}[x_l] = 0.5 \cdot (1 + 1) = 1$. 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.

IV. 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 [4] for the forward variance and a modification that looks at the gradients instead of the activations for the backwards case.

We employ the ResNet-50 and ResNet-101 architectures to extract the plots.

In particular, we extract the plots follows.

1) Classical ResNet with He initialization [13]; we considered both *fan-in* mode, i.e., (1), and *fan-out* mode - (2).
2) Same as the preceding with batch normalization.
3) ResNet with the three proposed residual summation modifications and their proper initialization to preserve the backwards variance.[1]
4) Same as the preceding but employing the initialization from [4].

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.

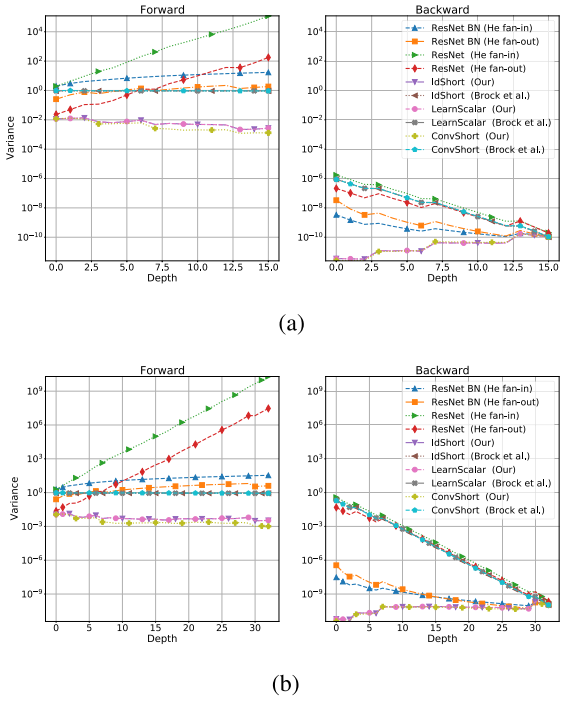
From Fig. 2, we first note that, as already pointed out in [4], classical ResNets with He initialization [13] do not preserve neither forward nor backwards signals while the use of batch normalization manages to fix things up. Interestingly, we note that the observed trends are more conspicuous in deeper networks.

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 in [4] 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 from [4] 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 [19] 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,

[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 $f(x)$.



(a)

(b)

Fig. 2. Signal propagation plots representing the variance of the forward activations (on the left) and the backward gradients variance (on the right) under different initialization schemes: both values refer to residual block output. Values on the x-axis denote the residual layer depth, while on the y-axis, the variance of the signal is reported in a logarithmic scale. (a) ResNet-50. (b) ResNet-101.

and a batch size of 128 (100 for ImageNet), in combination with a Cosine Annealing scheduler [22] that decreases the learning rate after every epoch. Moreover, in addition to the standard data augmentation techniques, we have also employed the recently proposed RandAugment method [6] and, for ImageNet only, the Label Smoothing technique [29].

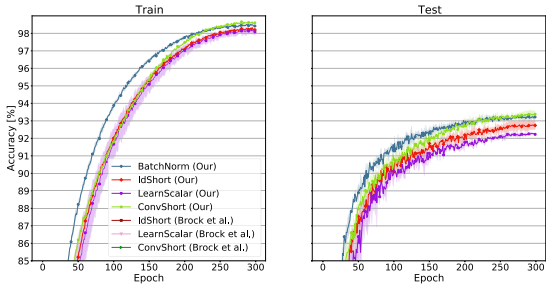
In Fig. 3, both train and test accuracies are shown for all the configurations. The results report the mean and the standard deviation of three independent runs.

The first thing to notice is that with the initialization scheme of [4], we are unable to train the network (the curve is actually absent from the plot) for both ResNet-50 and ResNet-101. 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. ResNet-18 can be trained using all the considered initialization (see Appendix B).

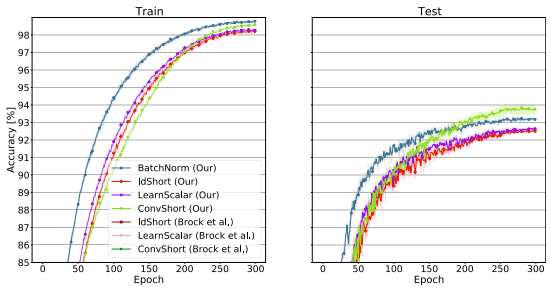
On the contrary, we can see how, thanks to the correct preservation of the backward signals, training is possible for all the proposed schemes when a gradient preserving initialization scheme is employed.

We also 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 Fig. 3(a) and (b), *ConvShort* appears to be an architectural change that, in combination with the proposed initialization strategy, is able to close the gap with a standard preactivated ResNet with batch normalization (it achieves the second-best in ResNet-18, see Appendix B).

To confirm the effectiveness of the proposed method, we also considered more resource-intensive settings, where gradient clipping is expected to be necessary. In particular, we considered the well-known datasets CIFAR-100 [19] and ImageNet [9]. Based on the results



(a)



(b)

Fig. 3. Test and Train accuracies of ResNet on CIFAR-10 dataset under different combinations of residual block modifications and initialization: standard ResNet with BatchNorm and IdShort, LearnScalar, ConvShort using both [4], and our 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, on the x-axis, we report the epoch at the which the accuracy (in the y-axis) has been computed. (a) ResNet-50. (b) ResNet-101.

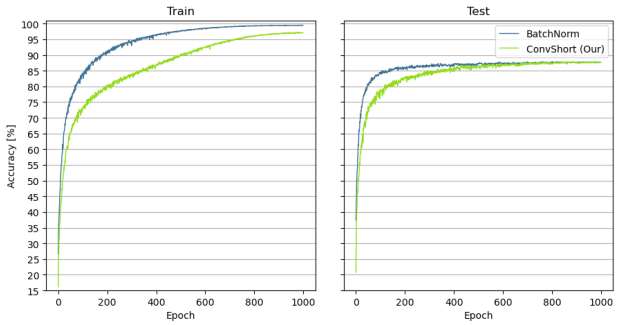


Fig. 4. Comparison of Train and Test accuracies of ResNet-50 between standard ResNet with BatchNorm and ConvShort with our initialization using CIFAR-100. Values on x-axis denote the epoch at the which the accuracy on the y-axis has been computed.

obtained with CIFAR-10, we decided to test the most promising among our architectures, namely, *ConvShort* modification.

In Fig. 4, we report the results obtained using our ShortConv modification and a standard ResNet-50 with batch normalization. As it is possible to see, training is slower for our setup, but the performance gap eventually closes and testing accuracy of our approach becomes even slightly superior at the end of the process. In Fig. 5, we show the results obtained with our ShortConv on the well-known ImageNet dataset. In order to evaluate the soundness of our proposal, we compare our results with the accuracy, reported on PyTorch [24], reached by a standard ResNet-50 trained on ImageNet. We can observe that the performance obtained with our architecture is in line with the state-of-the-art.

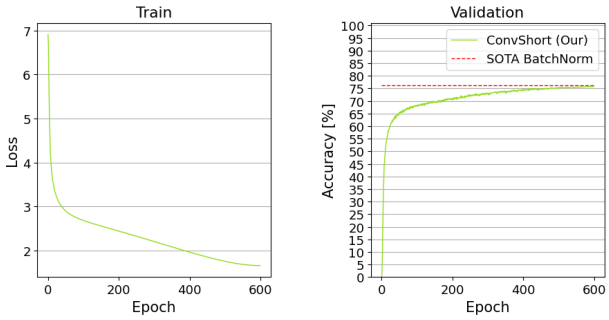


Fig. 5. Results obtained training ResNet-50 with our ConvShort modification on ImageNet. Values on x-axis denote the epoch at the which the accuracy on the y-axis has been computed. The red dashed line is the accuracy reported by PyTorch [24] for a standard ResNet-50 trained on ImageNet.

TABLE I
COMPUTATIONAL COST AND NUMBER OF PARAMETERS OF THE CONSIDERED ARCHITECTURES

Model	Input Resolution	Params (M)	#FLOPs (G)
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

The overall trend seems to indicate that DNNs can be trained up to state-of-the-art performance even without BN, even if this might come at the cost of a slightly longer training; moreover, a strong data augmentation might be needed to compensate the lack of the implicit regularization effects of BN.

To conclude, we report the number of parameters and FLOPs for the considered architecture in Table I. It is important to note that, despite *ConvShort* and *BatchNorm* having the same computational cost, our proposed method have some desirable characteristics (like the independence between the examples in a mini-batch). Moreover, the others configurations can be employed as more lightweight alternatives.

V. 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, CIFAR-100, and ImageNet. The obtained results are in line with the discussed theoretical analysis.

ACKNOWLEDGMENT

The authors would like to thank Dr. Soham De for kindly explaining to us some crucial aspects of his work. We would also like to thank Prof. Fabio Schoen for letting us work on this topic and putting at our disposal the resources of GOL and Prof. Andrew D. Bagdanov for his precious help in the refinement of this manuscript.

REFERENCES

[1] D. Arpit, V. Campos, and Y. Bengio, "How to initialize your network? Robust initialization for weightnorm & ResNets," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019.

[2] M. Awais, M. T. B. Iqbal, and S.-H. Bae, "Revisiting internal covariate shift for batch normalization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 5082–5092, Nov. 2021.

[3] T. Bachlechner, B. P. Majumder, H. H. Mao, G. W. Cottrell, and J. J. McAuley, "ReZero is all you need: Fast convergence at large depth," *CoRR*, vol. abs/2003.04887, pp. 1–14, Mar. 2020.

[4] A. Brock, S. De, and S. L. Smith, "Characterizing signal propagation to close the performance gap in unnormalized ResNets," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–30.

[5] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," 2021, *arXiv:2102.06171*.

[6] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2020, pp. 702–703.

[7] Y. N. Dauphin and S. Schoenholz, "MetaInit: Initializing learning by learning to initialize," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019.

[8] S. De and S. Smith, "Batch normalization biases residual blocks towards the identity function in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19964–19975.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[10] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[11] B. Hanin and D. Rolnick, "How to start training: The effect of initialization and architecture," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018.

[12] F. He, T. Liu, and D. Tao, "Why ResNet works? Residuals generalize," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5349–5362, Dec. 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[16] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: Closing the generalization gap in large batch training of neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 1729–1739.

[17] L. Huang, X. Liu, Y. Liu, B. Lang, and D. Tao, "Centered weight normalization in accelerating training of deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*. Venice, Italy: IEEE Computer Society, Oct. 2017, pp. 2822–2830.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[20] J. Lee, D. Joo, H. G. Hong, and J. Kim, "Residual continual learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 4553–4560.

[21] V. Lomonaco, D. Maltoni, and L. Pellegrini, "Rehearsal-free continual learning over small non-i.i.d. batches," in *Proc. CVPR Workshops*, 2020, pp. 989–998.

[22] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.

[23] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," 2019, *arXiv:1809.00846*.

[24] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035.

[25] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," 2020, *arXiv:1903.10520*.

[26] J. Shao, K. Hu, C. Wang, X. Xue, and B. Raj, "Is normalization indispensable for training deep neural network?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13434–13444.

[27] S. Wu et al., "L_1-norm batch normalization for efficient training of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 2043–2051, Jul. 2019.

[28] J. Yan, R. Wan, X. Zhang, W. Zhang, Y. Wei, and J. Sun, "Towards stabilizing batch statistics in backward propagation of batch normalization," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–17.

[29] C.-B. Zhang et al., "Delving deep into label smoothing," *IEEE Trans. Image Process.*, vol. 30, pp. 5984–5996, 2021.

[30] H. Zhang, Y. N. Dauphin, and T. Ma, "Residual learning without normalization via better initialization," in *Proc. Int. Conf. Learn. Represent.*, 2019, p. 2.

Open Access funding provided by 'Università degli Studi di Firenze' within the CRUI CARE Agreement