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 \( \alpha \) 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.

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)

\[
\z_l = \text{ReLU}(x_{l-1}), \quad x_l = W_l \z_l.
\]

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

\[
\text{Var}[W] = \frac{2}{n_m}
\]

(1)

where \( n_m = k^2c \) 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) \( \partial L / \partial x_l \).
architectural and algorithmic modifications that do 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 \( \alpha \) 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

\[ x_l = \alpha_l x_{l-1} + \beta_l f_l(x_{l-1}) \]

where coefficients \( \alpha_l \) and \( \beta_l \) 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)]:

\[ x_l = c \cdot (h(x_{l-1}) + f_l(x_{l-1})) \]

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

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

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

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

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

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

$$= c^2 \cdot (\text{Var}[h(x_{l-1})] + 1).$$

Thus, defining $h$ so that $E[h(x_{l-1})] = 0$ and $\text{Var}[h(x_{l-1})] = (1/c^2) - 1$, 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 $E[(\partial L/\partial x_l)] = 0$ and $\text{Var}[(\partial L/\partial x_l)] = C$ 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., $E[(\partial L/\partial x_l)] = 0$ and $\text{Var}[(\partial L/\partial x_l)] = C$. Since for the gradients at layer $l - 1$, we have

$$\frac{\partial L}{\partial x_{l-1}} = c \cdot \frac{\partial L}{\partial x_l} \cdot \frac{\partial x_{l-1}}{\partial x_{l-1}},$$

we get

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

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

$$\text{Var} \left[ \frac{\partial L}{\partial x_{l-1}} \right] = c^2 \cdot \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \text{Var} \left[ \frac{\partial h(x_{l-1})}{\partial x_l} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right]$$

$$+ \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] E \left[ \frac{\partial h(x_{l-1})}{\partial x_l} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right]^2$$

$$+ E \left[ \frac{\partial L}{\partial x_l} \right]^2 \text{Var} \left[ \frac{\partial h(x_{l-1})}{\partial x_l} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right]$$

$$= c^2 \left( \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \left( \text{Var} \left[ \frac{\partial h(x_{l-1})}{\partial x_l} \right] + \text{Var} \left[ \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right] \right) \right.$$  

$$+ \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] E \left[ \frac{\partial h(x_{l-1})}{\partial x_l} + \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right]^2$$

$$+ E \left[ \frac{\partial L}{\partial x_l} \right]^2 \left( \text{Var} \left[ \frac{\partial h(x_{l-1})}{\partial x_l} \right] + \text{Var} \left[ \frac{\partial f_l(x_{l-1})}{\partial x_{l-1}} \right] \right).$$

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

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

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

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 $\gamma_f^2 = 2/(1 - (1/\pi))$ in (3) breaks the conservation property of the gradients signal, as opposed to (1). As we back-propagate through the model, the factor $\gamma_f^2$ 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

$$\text{Var} \left[ \frac{\partial L}{\partial x_{l-1}} \right] = \frac{1}{2} \cdot \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \cdot 1 + \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \cdot 1$$

$$= \text{Var} \left[ \frac{\partial L}{\partial x_l} \right]$$

i.e., the variance of gradients is preserved. As for the activations, we get $E[x_l] = 0$ 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 [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): $h(x) = \alpha x$, $\alpha$ initialized at 1, $c = (0.5)^{1/2}$. In (6), we again get at initialization

$$\text{Var} \left[ \frac{\partial L}{\partial x_{l-1}} \right] = \frac{1}{2} \cdot \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \cdot 1 + \text{Var} \left[ \frac{\partial L}{\partial x_l} \right] \cdot \alpha^2$$

$$= \text{Var} \left[ \frac{\partial L}{\partial x_l} \right]$$
and similarly as above, we also obtain the forward preservation at all layers with $N = N$.

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

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

Again, if we consider the layers with equal size for inputs and outputs, we also get $E[x_l] = 0$ 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 $f(x_{l-1})$ 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 ComShort when $x$ has not the same pixel resolution or number of channels of $f(x)$.

---

**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.
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.

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.

TABLE I

| Model               | Input Resolution | Params (M) | #FLOPs (G) |
|---------------------|------------------|------------|------------|
| ResNet-50 BatchNorm | 32 × 32 × 3      | 38.02      | 4.2        |
| ResNet-50 IdShort   | 32 × 32 × 3      | 33.47      | 2.6        |
| ResNet-50 LearnScalar | 32 × 32 × 3   | 23.47      | 2.6        |
| ResNet-50 ConvShort | 32 × 32 × 3      | 38.02      | 4.2        |
| ResNet-101 BatchNorm | 32 × 32 × 3     | 74.78      | 8.92       |
| ResNet-101 IdShort  | 32 × 32 × 3      | 42.41      | 5.02       |
| ResNet-101 LearnScalar | 32 × 32 × 3    | 42.41      | 5.02       |
| ResNet-101 ConvShort | 32 × 32 × 3 | 74.78      | 8.92       |

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.

V. Conclusion

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.
