Exploring Normalization in Deep Residual Networks with Concatenated Rectified Linear Units

Wenling Shang
Oculus VR
wendy.shang@oculus.com

Justin Chiu
Facebook
justinchiu@fb.com

Kihyuk Sohn
NEC Labs America
ksohn@nec-labs.com

Abstract

Deep Residual Networks (ResNets) have recently achieved state-of-the-art results on many challenging computer vision tasks. In this work we analyze the role of Batch Normalization (BatchNorm) layers on ResNets in the hope of improving the current architecture and better incorporating other normalization techniques, such as Normalization Propagation (NormProp), into ResNets. Firstly, we verify that BatchNorm helps distribute representation learning to residual blocks at all layers, as opposed to a plain ResNet without BatchNorm where learning happens mostly in the latter part of the network. We also observe that BatchNorm well regularizes Concatenated ReLU (CReLU) activation scheme on ResNets, whose magnitude of activation grows by preserving both positive and negative responses when going deeper into the network. Secondly, we investigate the use of NormProp as a replacement for BatchNorm in ResNets. Though NormProp theoretically attains the same effect as BatchNorm on generic convolutional neural networks, the identity mapping of ResNets invalidates its theoretical promise and NormProp exhibits a significant performance drop when naively applied. To bridge the gap between BatchNorm and NormProp in ResNets, we propose a simple modification to NormProp and employ the CReLU activation scheme. We experiment on visual object recognition benchmark datasets such as CIFAR-10/100 and ImageNet and demonstrate that 1) the modified NormProp performs better than the original NormProp but is still not comparable to BatchNorm and 2) CReLU improves the performance of ResNets with or without normalizations.

Introduction

In recent years, convolutional neural networks (CNNs) have dominated many challenging computer vision tasks. From AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG (Simonyan and Zisserman 2014) to GoogleNet (Szegedy et al. 2015), there has been a gradual advance in the depth of convolution layers. The Deep Residual Network (ResNet) (He et al. 2016a) architecture, the latest break-through along this direction, scales up to hundreds or even thousands of layers yet still generate meaningful and improved representations by introducing skip connections, i.e., identity mappings, across several convolution layers. Following its emergence, a wealth of studies has been carried out to explain the compelling performance of ResNets (Veit, Wilber, and Belongie 2016), to understand the role of identity mappings (He et al. 2016b), to further deepen the networks (Huang et al. 2016) and to create wide but shallow alternatives (Zagoruyko and Komodakis 2016). In this work, we take a novel perspective and analyze another core ingredient of the success of ResNets, Batch Normalization (BatchNorm) (Ioffe and Szegedy 2015). We investigate the essential consequences of applying BatchNorm on ResNets with both ReLU activations (Nair and Hinton 2010) where gradients tend to vanish, and concatenated ReLU (CReLU) activations (Shang et al. 2016) where gradients tend to overly aggregate. Our discoveries show that BatchNorm stabilizes the scale of the network, thereby suppressing excessively big or small convolution responses, and that BatchNorm ResNets can further benefit from coupling with CReLU. Next we attempt to generalize the insights gained from BatchNorm to other types of normalization techniques because, as we will further discuss, there are circumstances when BatchNorm becomes less desirable. For an example, we choose Normalization Propagation (NormProp) (Arpit et al. 2016) as a BatchNorm substitute on ResNet. Our experimental and theoretical findings demonstrate that in order for NormProp ResNets to be more comparable to the baseline BatchNorm ResNet, it is beneficial to make appropriate adjustments to the original NormProp formulation and to use CReLU. Overall, our studies suggest that by better comprehending the nature of a technique on deep networks, we can potentially improve the existing method as well as design appropriate alternatives when in need.

Related Works

This section introduces the following key components in our work: Residual Networks, Batch Normalization, Normalization Propagation, and Concatenated ReLU.

Deep Residual Networks

Residual Networks (ResNets) (He et al. 2016a) are composed of stacked *residual blocks* and each block can be mathematically characterized as the following:

$$x_{l+1} = \mathrm{ReLU}\left(x_l + f(x_l, \mathcal{W}_l)\right),$$

where the function f consists of convolution, BatchNorm, and ReLU layers (Figure 1(a)).

Very deep ResNets have won over previous state-of-the-art on several tasks on ImageNet (Russakovsky et al. 2015) and MS COCO (Lin et al. 2014) by a significant margin. Features extracted from pre-trained ResNets have shown prominent results when transferred to other vision tasks (Ilievski, Yan, and Feng 2016). The concept of residual blocks has also influenced the design of other deep neural networks (Oord, Kalchbrenner, and Kavukcuoglu 2016). The great success of ResNets has sparked an abundance of follow-up research in order to decipher its intriguing properties and to improve the baseline architecture: Veit, Wilber, and Belongie interpret deep ResNets as exponential ensembles of shallow networks; Zagoruyko and Komodakis present the advantages of shallower but widened ResNets; Shah et al. replace ReLU with Exponential Linear Unit (ELU) (Clevert, Unterthiner, and Hochreiter 2015); Szegedy, Ioffe, and Vanhoucke combine residual blocks with inception; Huang et al. regularize extremely deep ResNets with stochastic layer dropout; He et al. search for the best activation ordering inside a residual block. In contrast with other existing works, we focus on exploring normalization techniques on ResNets.

BatchNorm and NormProp

Ioffe and Szegedy introduce Batch Normalization, which calculates the mean and standard deviation for each convolution filter response across each mini-batch at each iteration to normalize the current layer activation before advancing to the next layer. Mathematically speaking, for the filter matrix $\mathcal{W}^1$ from the l-th layer,

$$x_{l+1} = \left(\mathcal{W}_l x_l - \mathrm{E}_{\omega_l}[\mathcal{W}_l \omega_l]\right) / \sigma_{\omega_l}(\mathcal{W}_l \omega_l).$$

Note that, thanks to the mean subtraction, the bias term can be discarded from the convolution layer. BatchNorm alleviates the "internal covariate shift" phenomenon, where the distribution of x_l can fluctuate if not properly normalized. In practice, BatchNorm yields faster convergence and lifted performance. It also allows less stringent weight initialization and a larger starting learning rate. BatchNorm is especially meaningful to ResNets because of its depth and identity mappings, which we will elaborate in the **BatchNorm in ResNets** section.

Despite the undeniable value of BatchNorm, it also encounters certain drawbacks, such as non-negligible computation and memory overhead, as well as incompatibility with online learning (a batch size of 1) and recurrent neural networks (Ba, Kiros, and Hinton 2016). Several new normalization methods inspired by BatchNorm have been proposed to address the above issues. For example, Normalization Propagation (NormProp) (Arpit et al. 2016) and its close variant Weight Normalization (Salimans and Kingma 2016) reparameterize the weights instead of activations; Layer Normalization (Ba, Kiros, and Hinton 2016) assimilates BatchNorm procedures but obtain the statistics for each filter via a single example. Nevertheless, there has not been any documented attempt to integrate them into the state-of-the-art ResNets.

[1]Without loss of generality, we present for the case of fully connected layers only.

Among these techniques, NormProp is primarily tested on CNNs for vision tasks. We therefore take NormProp as an example and attempt to answer these questions in the **NormProp in ResNets** section: can NormProp achieve as competitive results as BatchNorm on ResNet? If not, why and how to shrink the gap?

Concatenated Rectified Linear Units

One of the most common non-linearities for deep neural networks is the Rectified Linear Units (ReLU) (Nair and Hinton 2010) which simply erases any negative responses, mathematically defined as $[\cdot]_+ \triangleq \max(\cdot, 0)$. Despite its popularity, several disadvantages of the ReLU nonlinearity have been observed, such as convergence slowdown (Maas, Hannun, and Ng 2013), filter redundancy (Shang et al. 2016) and bias shift (Clevert, Unterthiner, and Hochreiter 2015). A family of non-saturated ReLU alternatives, such as Leaky ReLU (Maas, Hannun, and Ng 2013), PReLU (He et al. 2015), ELU (Clevert, Unterthiner, and Hochreiter 2015), assign a small amount of activation to the negative responses instead of completely zeroing them out. Concatenated ReLU (CReLU) (Shang et al. 2016) is a remedy of another flavor–it concatenates the original linear responses and their negative copies then applies ReLU to both.

Definition 1. CReLU*, denoted by* $\rho_c : \mathbb{R} \to \mathbb{R}^2$*, is defined as:* $\rho_c(x) \triangleq ([x]_+, [-x]_+)$.

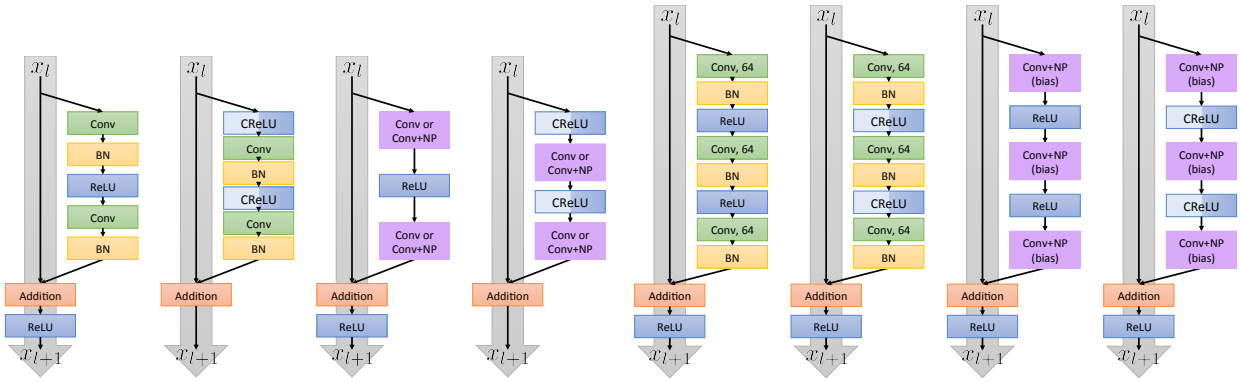
While ReLU's excessive sparsity can hamper optimization in latter layers, CReLU faces the opposite problem: its magnitude of activation grows when going deeper into the network (section **BatchNorm in ResNets**). Hence, normalization is of great necessity when integrating CReLU into very deep ResNets. However, Shang et al. only applies CReLU on relatively shallow CNNs without residual connections. Thus it is worth investigating how BatchNorm and NormProp impact the functionality of CReLU on ResNets.

It is relatively straightforward to replace ReLU with another activation function, as done by Shah et al.. However, CReLU alters the network architecture by adding additional output channels. It is unclear how to optimally incorporate CReLU activations into ResNets. In our design, we engage CReLU inside each residual block, avoid changing the identity path and keep the same number of convolution filters (Figure 1(a)). For each bottleneck blocks, we only add CReLU after the first two convolution layers (Figure 1(b)).

Batch Normalization in ResNets

Before BatchNorm was proposed, training state-of-the-art deep CNNs, such as AlexNet, VGG, GoogleNet, required careful initialization and more iterations. The deepest network then was generally based on GoogleNet (22 layers).

ResNets leverage identity mappings across every few convolution layers to delve as deep as hundreds of layers and achieves new state-of-the-art. Besides the identity mapping, BatchNorm is another indispensable ingredient in the success of ResNets. The increased depth of ResNets intensifies the importance of using BatchNorm. He et al. points out that deeper networks face many more optimization hurdles. BatchNorm layers can partly alleviate such difficulties. However, the identity mapping challenges the traditional



(a) Basic residual block architecture. From left to right, BatchNorm + ReLU, BatchNorm + CReLU, Plain/NormProp + ReLU, Plain/NormProp + CReLU.

(b) Bottleneck residual block architecture. From left to right, BatchNorm + ReLU, BatchNorm + CReLU, Plain/NormProp + ReLU, Plain/NormProp + CReLU.

Figure 1: Various residual blocks with different activation schemes and normalization techniques.

training strategies for deep CNNs. For example, most initialization techniques (Glorot and Bengio 2010; He et al. 2015; Mishkin and Matas 2016) target at generic convolution layers and much of the theoretical motivation of these techniques, such as making the expectation and variance of each layer's outputs to be 0 and 1, breaks down due to the identity mapping. In this case, BatchNorm enables training to happen even when it is hard to satisfy the ideal initialization conditions. To explore the effectiveness of different normalization techniques, we turn our attention to a ResNet-specific metric: the differences between the input to a residual block and its output. Specifically, if $x_l \in \mathbb{R}^{c\times w\times h}$ is an input to a residual block with c channels and $w \times h$ spatial dimension and $x_{l+1} = f(x_l, \mathcal{W}) + x_l, x_{l+1} \in \mathbb{R}^{c\times w\times h}$ is the output, we measure the difference by computing:

$$\Delta = \frac{1}{w \times h} \sum_w \sum_h \|x_l(:, i, j) - x_{l+1}(:, i, j)\|_2.$$

A large Δ indicates a dramatic change induced by the residual path and negligible contribution from identity path. A close-to-zero Δ indicates that information is mostly passed through the identity path only. We measure Δs across all 54 residual blocks in several 110-layer ResNet models trained on CIFAR-100, namely, plain ReLU/CReLU models and BatchNorm ReLU/CReLU models. It is worthwhile explaining our interest in CReLU models. CReLU equally preserves negative and positive activation as well as the corresponding gradients, thus the gradients of a CReLU network can easily aggregate if not properly normalized. As a result, the magnitude of activation can grow excessively when going deeper. ReLU networks, on the other hand, present a different optimization challenge–the activation gets excessively sparse due to zeroing out negative linear responses.

Δ's in Figure 2(a) displays a perhaps expected yet still informative trend. For both plain models, hardly any Δ occurs in the first 36 residual blocks (Figure 2(b), green and yellow bars), except for the spike when the spatial dimension is reduced; but during the latter half of the plain models, the values of Δ abruptly experience a huge jump onto a different order of magnitude. Differing from CReLU, whose Δs

Norm	Activation	Mean	NZR
Plain	ReLU	30.81 ± 36.69	0.14
NormProp		28.68 ± 33.58	0.21
BatchNorm		3.77 ± 4.43	0.51
Plain	CReLU	24.53 ± 26.02	0.50
NormProp		10.41 ± 10.28	0.50
BatchNorm		1.94 ± 1.99	0.50

Table 1: The mean activation (Mean) after last ReLU (i.e., before softmax layer) and the non-zero entries ratio (NZR). Note when using CReLU, the NZR is automatically 0.5.

aggregate gradually after 36th block as a result of its aforementioned nature, Δs for ReLU plain model almost shoot up exponentially.

By a huge contrast, for both BatchNorm models, the range of their Δs settles at around 1 to 5 for all residual blocks. An immediate conclusion is that BatchNorm induces consistent changes in activation throughout all layers of the network despite its extreme depth, but without BatchNorm the first half of the network is hardly utilized. In other words, discarding BatchNorm diminishes the potential of learning better representations using deeper models.

This observation also implies that BatchNorm encourages deep ResNets to sustain a stable, non-exploding activation "magnitude", whereas the plain networks suffer from dramatic, unstable increments in activation scales during the second half. We can further confirm this assertion by examining the mean of the last layer activation (after ReLU) before going into Softmax. Table 1 shows that plain networks produce activation with large mean and high variance while BatchNorm networks, including CReLU model (recall its vanilla version often ends up with large responses), produce activation with small mean and small variance. Maintaining a set of layer-wise consistent, non-exploding, non-vanishing model parameters is conjectured by Neyshabur, Salakhutdinov, and Srebro to be an important regularizer for deep networks. The intuition is that, even when two models are "scale equivalent"–meaning their output only differs by a multiplication scalar, the model whose parameters vary too much layer by layer undergoes unbalanced gradient updates across layers, while the one with parameter magnitude being consistent throughout the network enjoys more steady

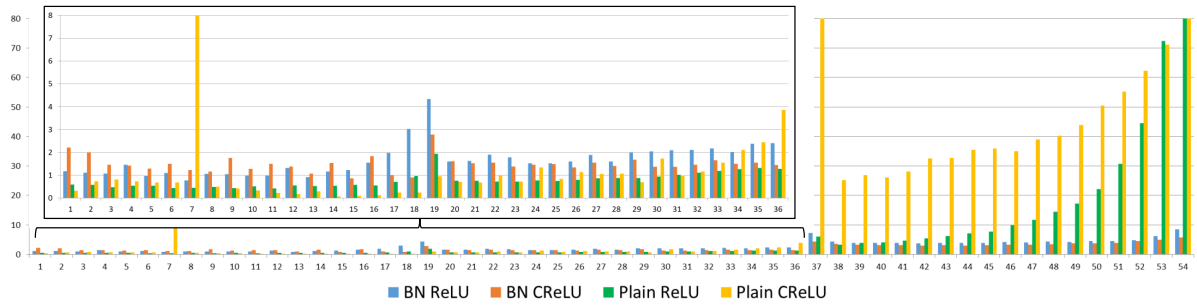


Figure 2: Input-output difference of each residual block for 110-layer ResNet trained on CIFAR-100 with or without BatchNorm using ReLU or CReLU. We zoom-in the differences for the first 36 residual blocks and provide a plot inside the black box. Best viewed in color on monitor.

and effective optimization. On this end, ResNets leverages BatchNorm to effectively maintain stable activation magnitude across all layers. Also, because BatchNorm redistributes the outputs of each layer to have zero mean and unit variance, the last layer still activates approximately 50% of the neurons after ReLU. By contrast, plain ReLU model only has 14% non-zero activation, which can result in overstretching the magnitude of activated entries, loss in representation power and inefficiency in backpropogation.

Finally, we would like to point out that, though BatchNorm ReLU and CReLU models demonstrate similar qualitative phenomenon in their Δs, replacing ReLU with CReLU consistently gives better recognition accuracy, under the same set of hyperparameters and training scheme. Especially for ImageNet ResNet models of different depths, CReLU activation demonstrates non-trivial improvement upon the baseline performance, which already is state-of-the-art.

Normalization Propagation in ResNets

In this section, we present both theoretical analysis and empirical evidence to show how the original formulation of NormProp fails in ResNets. Then, we provide two partial remedies by modifying the original formulation and by using CReLU activation scheme.

The NormProp formulation is given as follows:

$$x_{l+1} = \left[\frac{1}{2} - \frac{1}{2\pi}\right]^{-\frac{1}{2}} \left[\mathrm{ReLU}\left(\frac{\gamma_l(\mathcal{W}_l x_l)}{\|\mathcal{W}\|_2} + \beta_l\right) - \frac{1}{\sqrt{2\pi}}\right]$$

where β_l , $\mathcal{W}_l$, and γ_l are learnable parameters. Under the assumption that $\mathcal{W}$ has incoherent rows (i.e., filters) and input x_l has zero mean and unit variance, NormProp guarantees the output x_{l+1} to also approximately have zero mean and unit variance as well. However, we argue that the theoretical property of NormProp severely breaks down with residual connections and the original formulation of NormProp is difficult to be directly adopted for the following reasons:

1. ResNet convolution filters are **not** "incoherent", which contradicts one of the important assumptions for NormProp's theoretical backbone, *Canonical Error Bound*.
2. The identity mapping complicates the statistics of the output distribution, which is guaranteed by NormProp for generic CNN architectures. This cannot be easily fixed by a constant affine transformation since the variance after addition depends on the convolution filter weights.

We provide detailed illustrations regarding 1 and 2 in the supplementary materials. Moreover, as shown in Table 2, we verify that the naive application of NormProp on ResNet in replacement of BatchNorm significantly fails and can only be marginally better than plain ResNet.

Our initial modification to NormProp removes the constant affine transformation terms because they no longer help adjust the distribution of the output to have zero mean and unit variance, but rather complicate the learning by inducing meaningless negative activations. Mathematically, our modification is written as follows:

$$x_{l+1} = \left[\mathrm{ReLU}\left(\gamma_l(\mathcal{W}_l x_l)/\|\mathcal{W}_l\|_2 + \beta_l\right)\right]$$

and we denote our modified version as NormProp2. Note that NormProp2 embodies the same formulation as WeightNorm. Though NormProp2 is superior to NormProp, we still find it incompetent when applying to ImageNet, a large scale dataset. Thus, furthermore, we fix the multiplication parameter γ to derive NormProp3, especially targeting at ImageNet experiments. In this case, γ's are treated as hyperparameters and we set $\gamma = 1/1.21$, which is suggested as an initial value in the original NormProp (Arpit et al. 2016).

In addition to the aforementioned change, we propose to replace ReLU into CReLU for the following reasons:

1. CReLU can alleviate the vanishing activation issue of ReLU by keeping both positive and negative responses when competent normalization methods are lacking.
2. Since adding bias to plain ReLU ResNets delivers a much more positive impact than their CReLU counterparts (Table 2), we conjecture that CReLU activation is more robust to mean shift than ReLU, which is lacking in NormProp due to the identity path in ResNets.

Experiments

We validate our claims on visual object recognition benchmarks, namely CIFAR-10 and 100 (Krizhevsky 2009) and ImageNet (Deng et al. 2009). CIFAR-10 and 100 datasets are made of $50,000$ training and $10,000$ testing examples of 32×32 images evenly drawn from 10 and 100 classes, respectively. We choose 110-layer ResNet (He et al. 2016a) as our baseline and report the median of 5 runs. ImageNet dataset is composed of 1.3M images for training and $50,000$ for validation from $1,000$ object categories. In this experiment, we take 34-layer and 50-layer ResNet (He

et al. 2016a) as our baseline models. While the 34-layer model uses the standard building block as in Figure 1(a), the 50-layer ResNet utilizes the bottleneck block as in Figure 1(b) (He et al. 2016a).

Our implementation is based on Facebook ResNet implementations (Gross and Wilber 2016) and we reserve the data preprocessing and augmentation as well as hyperparameter values except for the initial learning rate and batch size for ImageNet. The default initial learning rate is 0.1, but we decrease it by half until training is feasible when the initial rate is too high. For imageNet experiments, we decrease the batch size from 256 to 128 due to computation consideration; to compensate the batch size reduction, we run the experiments for 150 epochs. Furthermore, we use MSR initialization (He et al. 2014) for models with BatchNorm, while other models are initialized with Xavier initialization (Glorot and Bengio 2010). Finally, we adapt the same shortcut connection as Gross and Wilber.[2] The summary results are provided in Table 2 and 4.

CReLU on ResNets

We observe a consistent improvement of CReLU models over their ReLU counterparts. Specifically, CReLU allows very deep networks to be trained without BatchNorm more reliably and reach fairly close to the best performance of ReLU models with BatchNorm on CIFAR-10 and 100. Though the CReLU activation scheme increases the model parameters by two, it should not be the major contributing factor as we can see in Table 2 that doubling the number of filters of ReLU models under *the same regularization setup* does not lead to a performance gain. To further demonstrate, we compare ReLU and CReLU models of similar size by reducing the number of filters for CReLU models. For CIFAR-10, ReLU model retains the baseline 16-32-64 design whereas the filters are reduced to 12-24-48 for CReLU model. CIFAR-100 undergoes underfitting with the baseline architecture but overfitting when doubling the number of filters without additional regularization. Therefore, for CIFAR-100, we apply doubled number of filters for ReLU model (32-64-128), 1.5 times for CReLU (24-48-96) and increase the weight decay coefficient to 0.0005. The results comparing BatchNorm ReLU and CReLU models of comparable sizes on CIFAR datasets are summarized in Table 3.

It is also noteworthy that for ReLU models on both CIFAR-10 and 100, simply adding the bias to the plain network improves the performance significantly, but much less to CReLU ResNet models. We contemplate that CReLU models are more robust to mean shift than ReLU ones.

Normalization in ResNets

As expected, the performance of NormProp models are far behind from BatchNorm ones on ReLU ResNets. For example, NormProp ResNet models show 11.60 and 37.60 test error rates on CIFAR-10 and 100 datasets, respectively, and this is merely better than those of plain ResNet with no bias and are not competitive to those of BatchNorm, which are

[2]In the case of NormProp, BatchNorm is applied on the identity path when spatial dimension is reduced to enable stable training.

Norm	Activation	CIFAR-10	CIFAR-100
Plain	ReLU	7.78	36.10
(no bias)		12.16	40.00
NormProp		11.60	37.60
NormProp2		7.46	31.71
BatchNorm		6.35	27.46
(double)		7.27	27.33
Plain	CReLU	6.86	28.24
(no bias)		7.17	28.71
NormProp2		7.22	29.29
BatchNorm		**5.72**	**25.89**

Table 2: Test error rates on CIFAR-10 and 100 datasets with different normalization methods including plain (with and without bias), NormProp, and BatchNorm and different activation scheme (ReLU, CReLU) based on 110-layer ResNet.

Activation	CIFAR-10 (size)	CIFAR-100 (size)
ReLU	6.35 (16-32-64)	22.50 (32-64-128)
CReLU	**5.87 (12-24-48)**	**22.06 (24-48-96)**

Table 3: Comparison between ReLU and CReLU BatchNorm ResNet models with similar number of model parameters after increasing the weight decay coefficients.

6.35 and 27.46. We note that the training error for NormProp models are not as low as BatchNorm models, hence overfitting is not the cause. More likely, the invalidation of the theory behind NormProp due to the identity mapping in ResNets is responsible for the unsatisfactory performance. Alternatively, NormProp2 (where we discard constant terms from the original formulation) demonstrates more competitive performance than the original NormProp and gets closer to the performance of BatchNorm ResNet models.

When it comes to CReLU activation scheme, however, neither NormProp nor our proposed variations are effective. In fact, the training significantly breaks down with NormProp and we did not manage to train CReLU ResNet models even after trying out many different initial learning rates. NormProp2 allows the model to be trained, but it is marginally effective given already a good performance of CReLU ResNet model without any normalization method in the case of CIFAR datasets. On the other hand, BatchNorm can be adopted smoothly to CReLU ResNet models, showing substantial improvement on both CIFAR-10 and 100 datasets.

Comprehensive experiments with different normalization methods on ImageNet dataset is conducted using 34-layer and 50-layer ResNet models. ImageNet dataset has much more training examples and output classes than CIFAR-10 or 100 datasets. Therefore the conclusions made from CIFAR datasets do not always carry over to ImageNet. Our results on ImageNet are mostly consistent with the observations we made previously. CReLU activation consistently shows superior performance to ReLU and 101-layer BatchNorm ResNet with CReLU achieves the best performance of 20.63 top-1 and 5.35 top-5 error rates. Our modified NormProp2 allows ResNet models to be trained but improvement over plain ResNet is not significant and is far behind the performance of models trained with BatchNorm. For NormProp3 models we fix all γ_i's during training which leads to improved performance, lowering the top-1 error rates of 34-layer CReLU ResNet model from 27.57 to

Depth	Act.	Norm	top-1	top-5	top-1†	top-5†
34	ReLU	Plain	29.92	10.84	27.15	9.16
		NP2	29.23	10.29	27.46	9.07
		NP3	29.72	10.54	27.38	9.16
		BN‡	26.73	8.74	24.76	7.35
	CReLU	Plain	28.85	10.28	26.59	8.90
		NP2	29.31	10.29	27.57	9.15
		NP3	28.28	9.84	26.26	8.73
		BN	25.62	8.28	23.72	7.15
50	ReLU	Plain	31.48	12.27	29.31	10.54
		NP2	29.75	10.80	28.16	9.79
		NP3	28.08	9.74	26.01	8.38
		BN‡	24.01	7.02	22.24	6.08
	CReLU	Plain	28.00	9.58	26.51	8.65
		NP2	29.06	10.31	27.58	9.35
		NP3	26.12	8.51	24.27	7.40
		BN	23.03	6.67	21.59	5.69
101	ReLU	BN‡	22.44	6.21	21.08	**5.35**
	CReLU	BN	**21.89**	**5.92**	**20.63**	**5.35**

Table 4: Error rates on ImageNet validation set on ResNets with different normalization techniques and depths. The top-1† and top-5† results are 10-crop results. The numbers with ‡ are taken from Gross and Wilber.

26.26. NormProp3 with CReLU has also shown improved performance over NormProp2 on 50-layer ResNet. However, there is still a non-trivial performance gap between the models with the variants of NormProp and BatchNorm.

Conclusion

We explore the essential role of BatchNorm in deep ResNets with two kinds of activation schemes, ReLU and CReLU which provide two different optimization landscapes. Motivated by our observations, we attempt to substitute BatchNorm with another normalization technique, namely NormProp. However, the intrinsics of the ResNet architecture makes the adaption non-trivial. Despite the challenge exhibited in ResNet modification, our work demonstrates that by maneuvering NormProp ResNet's design towards the direction that leads to similar effects of BatchNorm, the gap between BatchNorm and NormProp can be brought closer and that empirically CReLU is an effective add-on to both BatchNorm, to improve upon state-of-the-art performance, and NormProp, to render more comparable performances to BatchNorm baselines. Venues for future research involve transferring our findings to other neural net frameworks, such as recurrent neural networks, with other type of layer architectures.

Supplementary Materials

Background. We start with a recap of the theoretical premise upon which NormProp for generic CNNs is constructed. Arpit et al. initially motivate NormProp from the following important proposition, called *Canonical Error Bound*:

Proposition 1. (Canonical Error Bound) *Let $x_{l+1} = \mathcal{W}x_l$ where $x_l \in \mathbb{R}^n$, $x_{l+1} \in \mathbb{R}^m$ are random variables such that $\mathbb{E}_{x_l}[x_l x_l^T] = \sigma^2\mathbf{I}$ and the matrix $\mathcal{W} \in \mathbb{R}^{m\times n}$ is deterministic. Then the covariance matrix of x_{l+1} is approximately canonical, i.e., satisfying*

$$\min_{\alpha} \|\Sigma - \mathrm{diag}(\alpha)\|_{\mathrm{F}} \leq \sigma^2\mu\sqrt{\sum_{\mathrm{i,j=1;i\neq j}}^{\mathrm{m}} \|\mathcal{W}_{\mathrm{i}}\|_2^2\|\mathcal{W}_{\mathrm{j}}\|_2^2}. \quad (1)$$

where $\Sigma = \mathbb{E}[(x_{l+1} - \mathbb{E}[x_{l+1}])(x_{l+1} - \mathbb{E}[x_{l+1}])^T]$ is the covariance matrix of x_{l+1}, μ is the coherence of the rows of $\mathcal{W}$, $\alpha \in \mathbb{R}^m$ is the closest approximation of the covariance matrix to a canonical ellipsoid and $\mathrm{diag}(\cdot)$ converts a vector to a diagonal matrix. The corresponding optimal $\alpha_i^* = \sigma^2\|\mathcal{W}_i\|_2^2, \forall i \in \{1, \cdots, m\}$.

Recall that coherence μ is defined as

Definition 2. *The coherence of a matrix $\mathcal{W}$ is defined as*

$$\mu = \max_{i\neq j}(|\mathcal{W}_i^T\mathcal{W}_j|)/(\|\mathcal{W}_i\|_2\|\mathcal{W}_j\|_2).$$

Inequality (1) tells us that if $\mathcal{W}$ has almost orthogonal rows, i.e. $\mu \approx 0$, then Σ is almost diagonal. If assuming $\sigma = 1$, that is, each entry of x_l is independent with unit variance, then Σ is approximately the identity matrix. As a result, x_l, the output, has zero mean and unit variance. Extending the above consequence, Arpit et al. further assumes that the output x_{l+1} has standard Gaussian distribution. Under this additional assumption, after adding ReLU non-linearity, $\mathrm{ReLU}(x_{l+1})$ possesses the following distribution

Proposition 2. *Let $x \sim \mathcal{N}(0, 1)$, and $y = \mathrm{ReLU}(x)$, then $\mathbb{E}[y] = \frac{1}{2\pi}$ and $\mathrm{var(y)} = \frac{1}{2}(1 - \frac{1}{\pi})$.*

Therefore, in order for the output after ReLU non-linearity to be of zero mean and unit variance, the final formulation of NormProp is

$$x_{l+1} = \left[\frac{1}{2} - \frac{1}{2\pi}\right]^{-\frac{1}{2}}\left[\mathrm{ReLU}\left(\frac{\gamma_l(\mathcal{W}_l x_l)}{\|\mathcal{W}\|_2} + \beta_l\right) - \frac{1}{\sqrt{2\pi}}\right].$$

Coherence Assumption. Based on Proposition 0.1, in order for the linear output covariance matrix to be diagonal, it requires $\mu \approx 0$. However, in the case of ResNet, since identity mappings can also be responsible to pass on the information to the next block, intuitively $\mathcal{W}$ do not necessarily have almost orthogonal filters. To verify this conjecture, we empirically measure the coherence μ of each convolution filter matrix $\mathcal{W}_i$, $i = 1, \cdots, 34$, for the 34-layer NormProp2 ReLU ResNet trained on ImageNet (see Figure 3(a)). We also measure μ from NormProp2 CReLU ResNet since Shang et al. report the phenomenon of ReLU models exhibiting highly negatively-correlated filters. Indeed, neither CReLU nor ReLU NormProp ResNet has low-coherent convolution filter matrix. This finding substantially diminishes the validity of the important assumption from Proposition 0.1 and hence the theoretical promise of NormProp.

Identity Mapping. Now let us step back for a second and make a bold assumption that $\mathcal{W}$ is an orthogonal matrix. We also set up a simple residual block with $\mathcal{W}$ as shown in Figure 3(b). Though Proposition 0.1 approximates $x_{l+1} = \mathcal{W}x_l$ has zero mean zero and unit variance, by adding the identity path, $\mathcal{W}x_l + x_l$ no longer obeys the rule. Formally, we can compute the new statistics as following

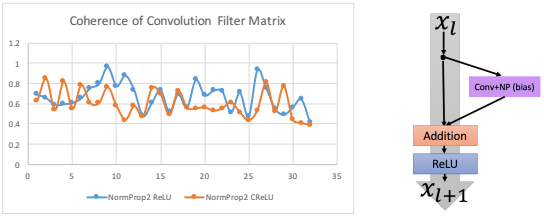


Figure 3: Left: coherence μ for each convolution filter matrix across 34 layers ResNets with NormProp2 trained on ImageNet. Right: an illustration of a simplified NormProp residual block.

Proposition 3. *Let $x_{l+1} = \mathcal{W}x_l + x_l$ where $x_l \in \mathbb{R}^n$ such that $x_l \sim \mathcal{N}(\mathbf{0}, I)$ and $\mathcal{W} \in \mathbb{R}^{n\times n}$ an orthogonal matrix. Then x_{l+1} is not a standard multivariate Gaussian random vector, that is $x_{l+1} \nsim \mathcal{N}(\mathbf{0}, I)$*

Proof. Since $x_{l+1} = \mathcal{W}x_l + Ix_l$ where x_l follows multivariate Gaussian distribution, then x_{l+1}, an affine transformation of x_l also follows multivariate Gaussian distribution. More precisely,

$$x_{l+1} \sim \mathcal{N}(\mathbf{0}, (\mathcal{W}+I)(\mathcal{W}+I)^T) = \mathcal{N}(\mathbf{0}, 2(\mathcal{W}+I))$$

Thus, the covariance matrix of x_{l+1} is not I. Moreover, the covariance matrix is dependent on $\mathcal{W}$. □

From Proposition 0.3, we can see that 1) the output of the residual block after addition no longer has identity variance and 2) its distribution cannot be adjusted to identity variance by any simple constant affine transformation as is done in the original NormProp formulation, because the covariance of the output is non-diagonal and dependent on $\mathcal{W}$, which is constantly changing during optimization.

Finally, for the theoretical promise does not hold under ResNet settings, it is justified to adapt NormProp formulation based on practical need. As further elaborated in the main text, we propose straightforward modifications (NormProp2 and NormProp3) which give better empirical performances than the original NormProp.

Acknowledgment

We sincerely thank Kaiming He for his helpful discussion.

References

Arpit, D.; Zhou, Y.; Kota, B. U.; and Govindaraju, V. 2016. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *ICML*.

Ba, J.; Kiros, J.; and Hinton, G. 2016. Layer normalization. *arXiv*.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2015. Fast and accurate deep network learning by exponential linear units. *ICLR*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-j.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Gross, S., and Wilber, M. 2016. Torch resnet blog. http://torch.ch/blog/2016/02/04/resnets.html.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. *ECCV*.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. 2016. Deep networks with stochastic depth. *ECCV*.

Ilievski, I.; Yan, S.; and Feng, J. 2016. A focused dynamic attention model for visual question answering. *arXiv*.

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

Krizhevsky, A.; Sutskever, l.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. *ECCV*.

Maas, A.; Hannun, A. Y.; and Ng, A. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop*.

Mishkin, D., and Matas, J. 2016. All you need is a good init. In *ICLR*.

Nair, V., and Hinton, G. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Neyshabur, B.; Salakhutdinov, R. R.; and Srebro, N. 2015. Path-sgd: Path-normalized optimization in deep neural networks. In *NIPS*.

Oord, A. v. d.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel recurrent neural networks. In *ICML*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*.

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*.

Shah, A.; Kadam, E.; Shah, H.; and Shinde, S. 2016. Deep residual networks with exponential linear unit. *arXiv*.

Shang, W.; Sohn, K.; Almeida, D.; and Lee, H. 2016. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Szegedy, C.; Ioffe, S.; and Vanhoucke, V. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR workshop*.

Veit, A.; Wilber, M.; and Belongie, S. 2016. Residual networks are exponential ensembles of relatively shallow networks. *arXiv*.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *BMVC*.