ERASERELU: A SIMPLE WAY TO EASE THE TRAINING OF DEEP CONVOLUTION NEURAL NETWORKS

Xuanyi Dong[1], **Guoliang Kang**[1], **Kun Zhan**[2], **Yi Yang**[1]
[1] CAI, University of Technology Sydney, [2] Lanzhou University

ABSTRACT

For most state-of-the-art architectures, Rectified Linear Unit (ReLU) becomes a standard component accompanied by each layer. Although ReLU can ease the network training to an extent, the character of blocking negative values may suppress the propagation of useful information and leads to the difficulty of optimizing *very deep* Convolutional Neural Networks (CNNs). Moreover, stacking of layers with nonlinear activations is hard to approximate the intrinsic linear transformations between feature representations. In this paper, we investigate the effect of erasing ReLUs of certain layers and apply it to various representative architectures. We name our approach as "EraseReLU". It can ease the optimization and improve the generalization performance for *very deep* CNN models. In experiments, this method successfully improves the performance of various representative architectures, and we report the improved results on SVHN, CIFAR-10/100, and ImageNet-1k. By using EraseReLU, we achieve state-of-the-art single-model performance on CIFAR-100 with 83.47% accuracy. Codes will be released soon.

1 INTRODUCTION

Since the success of AlexNet Krizhevsky et al. (2012) in the ILSVRC-2012 competition Russakovsky et al. (2015), more and more researchers move their focus on deep CNN models. Features learned from the neural networks successfully improve the performance of the large-scale vision recognition task, and can be transferred to a large variety of computer vision tasks, such as object detection Girshick et al. (2014), pose estimation Wei et al. (2016) and human-object interactions Gkioxari et al. (2017). Given the powerful transfer ability, many researchers focus on the "network engineering" designing more effective and efficient network architectures.

The state-of-the-art CNN architectures become increasingly deeper and more complex. The VGGNet Simonyan & Zisserman (2014) extends the depth of AlexNet from eight to nineteen layers. The GoogleNet Szegedy et al. (2015; 2016) designs the Inception module explicitly by incorporating the multi-scale property into the architecture. ResNet He et al. (2016a) presents a residual learning framework to ease the training of networks, and successfully train a network with more than 1000 layers. DenseNet Huang et al. (2017) connects each layer to every other layer, which encourages feature reuse and substantially reduces the number of parameters. These different architectures share a key characteristic: they incorporate many nonlinear units, ReLU, in the networks.

ReLU layers are widely used in all CNN architectures. It has also been demonstrated to be more powerful than some other nonlinear layers in most situations, such as sigmoid and tanh Glorot et al. (2011). ReLU can not only increase the nonlinearity but also ameliorate gradient vanish/explosion phenomenon in CNNs. Therefore, a convolution layer or a fully-connected layer is usually accompanied with a ReLU layer by default in the state-of-the-art CNN architectures. *Is this design principle necessary and helpful for the classification or some other vision tasks?*

We found that reducing the nonlinearity in *very deep* CNNs eases the difficulty of neural network training. With the network going deeper, the benefits from depth and complex become less and less. For example, there is about 1.3% accuracy improvement from ResNet-110 to ResNet-164 on CIFAR-10 Krizhevsky & Hinton (2009). However, six times deeper network, ResNet-1001, even decreases the accuracy about 1.7%. In this way, the trained network is far from the capacity it should achieve. When we simply erase the last ReLU layer in residual blocks, the accuracy for a network

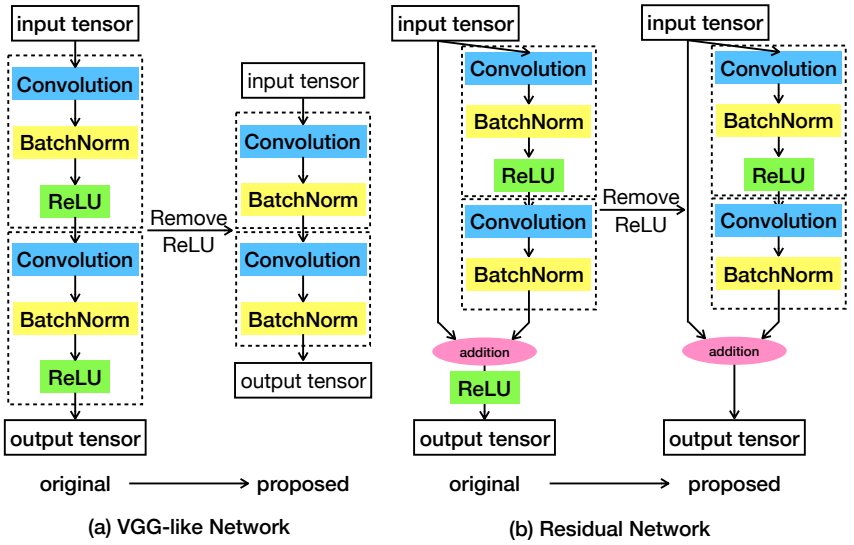


Figure 1: (a) shows a part of VGG-like networks. (b) shows a block of ResNet. "original" and "proposed" mean the original architecture and the improved architecture by erasing ReLU layers. Our proposed model usually achieves a higher performance compared to the original one.

with more than 1000 layers can still increase, whereas the *very deep* ResNet tends to decrease the accuracy.

In this work, we propose a simple but effective method to improve the performance of deep CNN architectures, which erasing ReLUs of a subset of layers in the deep neural network. We name our method "EraseReLU". For simplification, we denote the atomic module in CNN architectures as the basic unit, *e.g.*, the residual block in ResNet and Inception module in GoogleNet. Our EraseReLU simply erases the last ReLU layer in the basic unit. For example, there are three ReLU layers in the bottleneck residual unit, and EraseReLU will erase the last ReLU layer. Information through the forward pass might be suppressed by some nonlinear layers. For example, in some unusual situations Dong et al. (2017), the most of output after ReLU are zero and can not recover because the most of gradient are also zero. As our approach significantly reduces the nonlinearity in CNN models, it helps the information propagation in *very deep* networks. EraseReLU also benefits the network optimization, usually leading the model to converge faster in the early training epochs.

We empirically demonstrate that EraseReLU can improve the performance of various state-of-the-art CNN architectures. On four benchmark dataset (CIFAR-10, CIFAR-100, SVHN, and ImageNet-1k), we apply our EraseReLU on ResNet, Pre-act ResNet He et al. (2016b), Wide ResNet, Inception-V2 and ResNeXt. Most of these models with EraseReLU achieve higher classification results, and also converge faster than original models. Further, we significantly outperform the state-of-the-art result on CIFAR-100.

2 Related Work

2.1 Nonlinearity

The nonlinear unit plays an essential role in strengthening the representation ability of a deep neural network. In early years, sigmoid or tanh are standard recipes for building shallow neural networks. Since the rise of deep learning, ReLU Glorot et al. (2011) has been found to be more powerful in easing the training of deep architectures and contributed a lot to the success of many record-holders Krizhevsky et al. (2012); Simonyan & Zisserman (2014); Szegedy et al. (2016; 2017); He et al. (2016a). There exist lots of variants of ReLU nowadays, such as Leaky ReLU Maas et al. (2013), PReLU He et al. (2015), *etc*. The common ground shared by these units is that the computation will be linear on a subset of neurons. Models trained with such kinds of nonlinear units can be viewed as a combination of an exponential number of linear models which share parameters Nair & Hinton

(2010). This inspires us that modeling the local linearity explicitly may be useful. In this paper, we extend this linearity from the *subset* to the *full set* of neurons in some layer and empirically found that this extension effectively improves the performance of the model.

2.2 ARCHITECTURE

Krizhevsky won the ILSVRC-2012 competition and revealed that a large and deep CNN is capable of achieving benchmark results on a highly challenging dataset. Except for the AlexNet architecture Krizhevsky et al. (2012), they also demonstrate several techniques that are essential for training a good CNN model, such as ReLU, Dropout and data argumentation. Zeiler & Fergus (2014) *et al*. proposed a novel visualization technique providing insight into the CNN features as well as a new architecture ZFNet. NIN Lin et al. (2013) leverages one-by-one convolutional layers to make the network become deeper and yield better performance. VGGNet Simonyan & Zisserman (2014) utilizes 3×3 convolutional filters to replace the 5×5 and 7×7 filters in AlexNet, thus makes the network deeper (19 weight layers) to obtain a significant improvement.

Szegedy et al. (2015) *et al*. first proposed the GoogleNet architecture. It considers the Hebbian and multi-scale principle thus design the "Inception module" in the network. Ioffe & Szegedy (2015) *et al*. proposed the Batch Normalization to accelerate the network training. They also applied Batch Normalization to a new variant of the GoogleNet, named as BN-Inception. Szegedy et al. (2016) *et al*. proposed several general design principles to improve Inception module. Using these principles, they design a new CNN architecture, named as Inception-v3. It improves the ILSVRC 2012 classification results to 21.2% in terms of top-1 error. Szegedy et al. (2017) *et al*. combined the advantages of Inception architectures with residual connections to improve training speed for the Inception architecture. Kontschieder et al. (2015) *et al*. train classification trees with the representation learned from deep convolutional networks in an end-to-end manner.

Highway network Srivastava et al. (2015) is designed to ease gradient-based training of very deep networks. He et al. (2016a) *et al*. proposed the deep residual network that achieves a remarkable breakthrough in ImageNet classification and won the 1st places various of ImageNet and COCO competitions. The proposed residual learning can make the network easier to optimize and gain accuracy from considerably increased depth. Following ResNet, He et al. (2016b) proposed the Pre-activation ResNet improving the ResNet by using pre-activation block. Wide ResNet Zagoruyko & Komodakis (2016) decreases depth and increases the width of residual networks. It tackles the problem of diminishing feature reuse for training very deep residual networks. ResNeXt Xie et al. (2017) optimizes the convolution layer in ResNet by aggregating a set of transformations with the same topology.

3 METHODOLOGY

The mapping between different representations can be linear or non-linear. If the intrinsic relationship between two different representations is linear mapping, it is hard to learn to approximate such kind of linear mapping through stacking of non-linear transformations, especially when the architecture is quite deep and the data is scarce. He et al. (2016a) illustrates that optimization can be difficult in approximating an identity mapping by stacking multiple layers with non-linear activations in very deep neural networks. ReLU layers introduce linearity for the subset of neurons with positive responses, but we find that for a subset of layers, keeping linearity also for the neurons with negative responses, i.e. explicitly forcing the linear mapping for a subset of layers, can be helpful.

3.1 ERASERELU

Most architectures design a core module and stack multiple such modules with different configurations. Therefore, a network can be formulated [1] as :

$$F(x) = f_n \circ f_{n-1} \circ ... \circ f_i \circ ... \circ f_2 \circ f_1 \circ x \tag{1}$$

where each f_i indicates the $i-th$ basic unit in the network and $f_i \circ x$ equals $f_i(x)$. As illustrated in Fig. 2, f can be a convolutional layer followed by one ReLU layer, a residual block, and a Inception

[1] For simplification, we ignore the fully-connected layer and pooling layer.

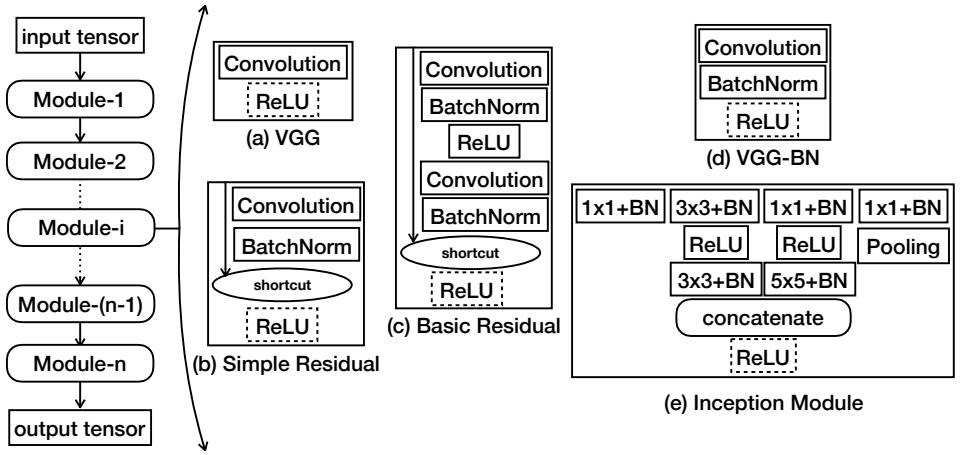


Figure 2: In most CNN architectures, the feature extraction part is stacked by many similar modules with different configurations. (a) and (d) are modules for VGG style network. (b) and (c) show the residual style modules. (e) shows an example of the Inception module. Our EraseReLU erases the ReLU with dashed line.

module for VGG, ResNet and GoogleNet, respectively. As the number of stacked module increases, the network tends to overfitting the training set and become difficulty to be optimized. Residual connection He et al. (2016a) alleviates this phenomenon to some extend, but these two problems are still unsolved. We empirically find that reducing nonlinearities of f are helpful for ameliorating the overfitting and optimization problems in *very deep* neural networks.

What is the most efficient way to reduce the nonlinearities of f and maintain the model capacity in the same time? It is obvious that there are usually three operations in different modules: convolution, batch normalization and ReLU. Convolution operation is a linear unit and also essential for model capacity, thus we can not erase the convolution operation. Batch normalization is not a linear unit strictly, but can be approximately regarded as a linear unit. It can avoid the gradient explosion by stabilizing the distribution and reducing the internal covariate, thus we should also retain this operation. Therefore, there left two ways to reduce the nonlinearities, modifying ReLU or optimizing the module structure. The module structure has thousands of combination of different operations, which is beyond the scope of our paper. Hereto, we eliminate all choices except modifying ReLU.

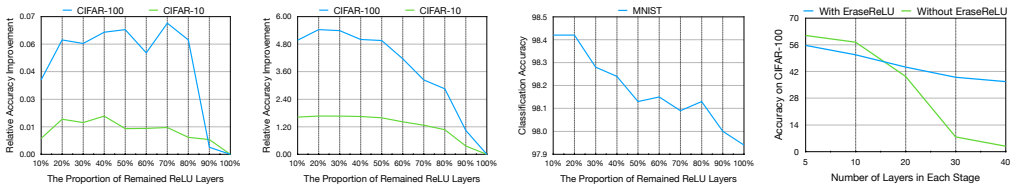
We can observe that modules shown in Fig. 2 share a common character, the last layer in these modules is a ReLU layer:

$$F(x) = Module \circ x = ReLU \circ Module^{'} \circ x \tag{2}$$

where most architectures also have this character. If we erase the last ReLU layer, the overall module structure can be reversed. On the contrary, if we erase ReLU in the middle part of these modules, it can destroy the module structure and decrease the module capacity, thus be harmful for the whole model performance. From experiments, we observe erasing this last ReLU layer in each module can be capable of easing the training difficulty and considered as a regularization improving the final performance.

There exist many variants of ReLU, *e.g.* Leaky ReLU and PReLU. They all alleviate the gradient vanishing problem. While these variants may benefit the training of deep neural networks compared to ReLU, the performance improvement is negligible Xu et al. (2015). Inspired by ReLU that inducing linearity for a subset of neurons, we explicitly inducing the linear mapping *i.e.* erasing ReLUs, for a subset of layers. Experiments illustrate that EraseReLU outperforms ReLU with a large margin.

Modules in Fig. 2 can be categorized as after-activation He et al. (2016b), where activation operations (Batch Normalization & ReLU) are after the convolutional layer. EraseReLU is suitable for this kind of modules. However, there exists another kind of modules, *i.e.* pre-activation. Pre-activation style is first introduced by Pre-act ResNet. Following He et al. (2016b), some other researchers also utilize this pre-activation style in their architectures, such Wide ResNet Zagoruyko & Komodakis



(a) VGG-style Network (b) Residual-based Network (c) Multiple Layer Perception Network (d) VGG-style Network

Figure 3: Comparison of classification accuracy with/without EraseReLU. In Fig.3(a), we show the relatively accuracy improvement of a VGG-style network with 31 weighted layers in terms of the different proportion of ReLU layers after applying EraseReLU. In Fig.3(b), we show the relatively accuracy improvement of a residual-based network with 31 weighted layers in terms of the different proportion of ReLU layers after applying EraseReLU. In these two figures, the green line and the blue line represent the results of CIFAR-100 and CIFAR-10, respectively. In Fig.3(c), we show the absolute accuracy comparison of a multiple layer perception network with 12 weighted layers on MNIST, in terms of the different proportion of ReLU layers after applying EraseReLU. In Fig.3(d), we show the accuracy on CIFAR-100 in terms of different number of weighted layers in the VGG-style network, which is the same as in Fig.3(a).

(2016) and DenseNet Huang et al. (2017). The location of ReLU in pre-activation modules is not the tail of module, thus EraseReLU may harm the pre-activation structure.

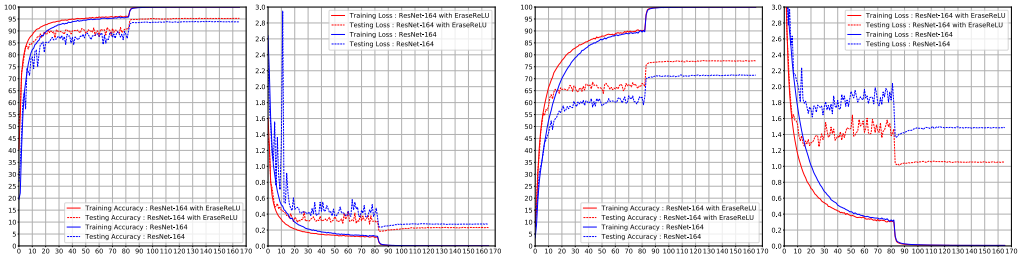
We empirically find that EraseReLU can yield a small improvement of these pre-activation modules on small scale dataset, *e.g.* CIFAR-10. For these models, we directly erase the last ReLU layer in each module. However, if we modify these pre-activation module into after-activation then applying EraseReLU, the performance can be better than the original pre-activation models as well as the pre-activation models with EraseReLU. For example, Wide ResNet will be modified as ResNet with EraseReLU with wider channel in each residual blocks. On large scale dataset, when applying EraseReLU on about $\frac{1}{3}$ or even $\frac{1}{2}$ pre-activation modules, we can still obtain similar results with faster convergence rate in the early training epochs.

3.2 Analysis of the EreaseReLU's Effect

In this section, we analyze the effect of the number of modules with EraseReLU. We perform two models stacked by 30 similar modules on CIFAR datasets. One model uses the VGG-style module (module b in Fig. 2), and the other uses the residual-style module (module c in Fig. 2). Both of them use a convolutional layer with $16 \times 3 \times 3 \times 3$ kernel size mapping the 32×32 input image into a $16 \times 32 \times 32$ feature tensor at the first layer. Then there are three stages following the first convolutional layer, where each stage has ten modules with the different number of output channels, *i.e.*, 16, 32 and 64, respectively. Between each stage, we use a max pooling layer with the kernel size of three to downsample the spatial size by stride equaling two. Therefore, each of these two models has 31 weighted layers. Moreover, we try a multiple layer perceptron (MLP) network with 12 fully-connected layers. Each fully-connected layer in this MLP has 1000 output neurons followed by BatchNorm+ReLU+Dropout, except the last one maps the 1000 input neurons to 10 neurons for MNIST classification.

We conduct the different number of modules to apply EraseReLU. For a certain number of modules to assign EraseReLU, we uniformly sample positions in 10 layers within each stage. We demonstrate the relative accuracy improvement[2] for VGG-style and residual-based network in Fig. 2. When the depth is 31, it is obvious that even a model with only 10% ReLU layers outperforms the model where each weighted layer is followed by ReLU. For VGG-style networks, this deep model is very difficult to optimize. There is a clear tendency that the model with less ReLU layers leads a better performance, and it achieves the highest accuracy when only using 20% ReLU layers. The residual-based network, shown in Fig. 3(b), greatly eases the training procedure, while it is still inferior to those with EraseReLU. Even the model with 4 ReLU layers outperforms that with 31 ReLU layers.

[2] $\frac{\text{Accuracy}_\text{E} - \text{Accuracy}_\text{O}}{\text{Accuracy}_\text{O}}$, E:EraseReLU and O:Original



(a) Accuracy/Loss of ResNet-164 on CIFAR-10 (b) Accuracy/Loss of ResNet-164 on CIFAR-100

Figure 4: Comparison of the accuracy and loss between ResNet-164 and ResNet-164 with EraseReLU. In each subfigure, the first picture shows the accuracy (%, y-axis) v.s. the training epoch (x-axis), and the second picture shows the loss value (y-axis) v.s. the training epoch (x-axis). The red line is the model using our EraseReLU, and the blue line is the original model. The solid and dashed lines show the performance of the training and testing data.

The weighted layers in MLP only contain fully-connected layer. From Fig.3(c), we can observe the similar phenomenon, less ReLU yields better performance.

We can see from the Fig.3(d) that the network trained with EraseReLU will gradually become better than the one without EraseReLU when the network goes deeper.

4 EXPERIMENTS

In this section, we investigate the performance of the proposed method. We empirically demonstrate the effectiveness of our proposed approach on a various of state-of-the-art architectures. The improved architectures are evaluated on four datasets, SVHN Netzer et al. (2011), CIFAR-10, CIFAR-100 Krizhevsky & Hinton (2009) and ImageNet-1k Russakovsky et al. (2015).

4.1 DATASETS

SVHN. The Street View House Numbers (SVHN) dataset is obtained from house numbers in Google Street View images. This dataset contains ten classes, from digit 0 to digit 9. There are 73257 32×32 colored digit images in the training set, as well as the 26032 images in the testing set. It also has 531131 additional, somewhat less difficult samples, to use as extra training data. Following the common experiment setting on SVHN, such as Zagoruyko & Komodakis (2016); Huang et al. (2017), we only use the official training and testing data. When training models, we only divide the pixel values by 255 to range into [0,1] without any data augmentation.

CIFAR. The CIFAR-10 dataset consists of 60000 32x32 colored images in 10 classes. There are 50000 training images with 5000 images per class, and 10000 testing images with 1000 images per class. The CIFAR-100 dataset is similar as CIFAR-10 but contains 100 classes. We use the official training and testing sets on these two datasets. Following the common practice He et al. (2016a); Xie et al. (2017); Huang et al. (2017), we normalize the data using the means and standard deviations of RGB channels. We also adopt the random horizontal flip and random crop[3] data argumentations.

ImageNet. The ILSVRC-1k classification dataset contains 1000 classes. There are about 1.2 million images for training, and 50000 for validation. We use the same data argumentation as in Xie et al. (2017); Huang et al. (2017) for training. For evaluation, we only use the single-crop with input image size 224224. We also generate three subsets of ImageNet for empirical studies. These three subsets randomly sample 10%, 20% and 30% images from the ImageNet dataset. Therefore, all of them have 1000 classes for training and testing images, but the number of images are different. We refer these three datasets as ImageNet-10%, ImageNet-20% and ImageNet-30%. To be noticed, ImageNet-10% is a subset of ImageNet-20%, and ImageNet-20% is a subset of ImageNet-30%.

[3]Pad 4 pixels on each border and randomly crop a 32x32 region

Architecture	Setting	CIFAR-10		CIFAR-100		SVHN	
		original	**EraseReLU**	original	**EraseReLU**	original	**EraseReLU**
ResNet	56	6.97	5.95	29.24†	27.75†	-	-
	110	6.43	5.71	27.95†	25.76†	-	-
	1202	7.93	4.96	-	-	-	-
Pre-act ResNet	110	6.37	5.71	27.20	25.76	-	-
	164	5.46	4.65	24.33	22.41	-	-
	1001	4.92	4.10	22.71	20.63	-	-
Wide ResNet	40-4	4.97	4.27	22.89	20.07	-	-
	28-10	4.00	3.78	19.25	19.10	-	-
	28-10-D	3.89	3.88	18.85	18.60	-	-
	52-1	6.43	5.37	29.89	26.45	2.08	1.87
	52-1-D	6.28	5.65	29.78	25.00	1.70	1.54
	52-10	3.69†	3.69	19.19†	18.19	1.81	1.64
ResNeXt	29-8X64	3.65	3.57	17.77	17.23	-	-
	29-16X64	3.58	**3.56**	17.31	**16.53**	-	-
Inception-V2	30	5.45†	5.29	28.29†	24.34	-	-

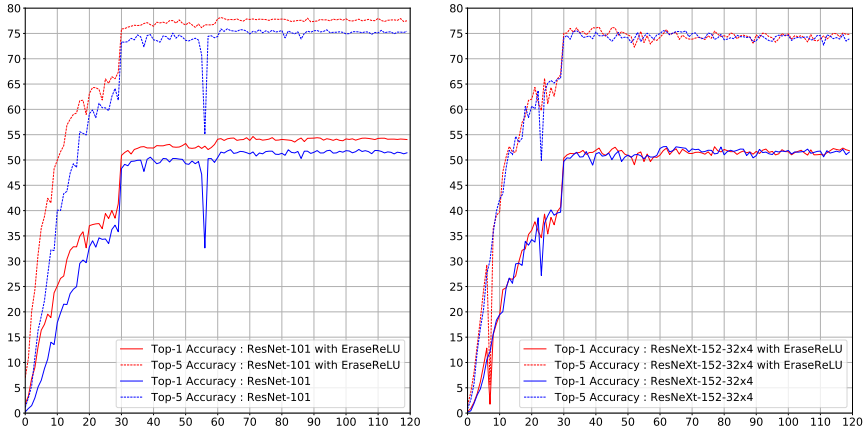
Table 1: Classification error (%) on CIFAR and SVHN datasets. The "original" means the architecture described in the original papers, and the EraseReLU means the architecture by applying our EraseReLU method on the original one. † indicates results run by ourselves. The setting formats follow the original papers, except that the setting of Inception-V2 indicates there are 30 Inception modules in the network. By applying EraseReLU, most of the architectures achieve lower error rates while using the same parameters and computation cost.

4.2 Experiments on SVHN and CIFAR

In this section, we apply our approach to improve five different architectures, ResNet, Pre-act ResNet, Wide ResNet, ResNeXt and Inception-V2 Ioffe & Szegedy (2015). We demonstrate the comparison on SVHN, CIFAR-10, and CIFAR-100.

Experiment settings. By default, we change the architecture to after-activation style for those using the pre-activation modules, and then apply EraseReLU. We apply EraseReLU on all residual modules for ResNet, Pre-act ResNet, Wide ResNet and ResNeXt. For Inception-V2, we apply EraseReLU on all half Inception modules, where the positions that applied EraseReLU are uniformly sampled. We use stochastic gradient descent (SGD) and a Nesterov momentum Sutskever et al. (2013) of 0.9 for all CNN models. The model with EraseReLU is trained in the same way as the original model. For ResNet and Pre-act ResNet , we follow the same training setting as in He et al. (2016a;b). The batch size is 128 on 2 GPUs. The initial learning rate is set to 0.1 and is divided by 10 at 82-th and 123-th epochs. For the network with more than 110 layers, we warm up the training by using a smaller learning rate of 0.01 at the first training epoch and go back to 0.1 after that. For Wide ResNet, we follow the same training procedure as in Zagoruyko & Komodakis (2016). Following Xie et al. (2017), we train ResNeXt on eight GPUs with a mini-batch size of 128 and weight decay of 0.0005. We start the learning rate of 0.1 and train the model for 300 epochs, reducing it at 150-th and 225-th epochs. As the Inception-V2 was designed for the ImageNet with 224×224 image size, we design a special Inception-V2 model for CIFAR datasets. It is shown as module e in Fig. 2, where the 5×5 convolutional layer is replaced by two 3×3 convolutional layers. The proportions of output channels from the 1×1, $3\times$, 5×5 convolution branches and the pooling branch are $1 : 8 : 2 : 1$ in one Inception module. Our Inception-V2 on CIFAR has a convolutional layer with 64 output channels at first, followed by three Inception stages. Each stage contains ten Inception modules, where the base output channels are 16, 32 and 64, respectively. The Inception-V2 is trained in the same way as Wide ResNet.

Analysis. We show the accuracy and loss curves of the original ResNet-110/ResNet-164 and the models using EraseReLU in Fig. 4. As can be observed, erasing ReLU layers in such deep neural networks can not only achieve higher accuracies but also reduce the overfitting of the training data. When the learning rate reduces from 0.1 to 0.01, we can observe the training loss goes down, but the testing loss goes up. It is caused by over-fitting. The loss of original ResNet-110 rises from



(a) ResNet-101 (b) ResNeXt-152 (32x4d)

Figure 5: Comparison of models with and without EraseReLU on the ImageNet-10% in terms of the top-1 and top-5 accuracy (%). The first subfigure show the results of ResNet-101, and the second one is from ResNeXt-152 (32x4d). The blue and red lines indicate the original network and the network using EraseReLU, respectively. The solid and dashed lines show the top-1 and top-5 accuracy, respectively.

1.2 to 1.5, while the model using EraseReLU just rises from 1.1 to 1.3. For the ResNet-164 model, by using EraseReLU the loss just rises about 0.01, but the original model raises more than 0.1. Therefore, the model using EraseReLU results in a much smaller loss increase than the original model. Moreover, with the network going deeper, we can obtain more benefits from EraseReLU. For example, the improvement of EraseReLU on ResNet-164 is much more than ResNet-110.

We demonstrate the main results on SVHN and CIFAR in Table 1. For ResNet, EraseReLU improves the accuracy compared to the original architectures. The ResNet with 1202 layers achieves a worse performance than the network with 110 layers, but ours can still maintain the accuracy improvement even when the depth becomes very deep. For Pre-act ResNet, we use both after-activation and EraseReLU. For Pre-act ResNet with 1001 layers on CIFAR-100, the error drops from 22.71% to 20.07% by our EraseReLU. We can observe more than 2% accuracy gain on Pre-act ResNet with 1001 layers. As shown in He et al. (2016b), the after-activation style does not improve the performance. Thus the main contribution of the accuracy improvement is from EraseReLU. For ResNeXt, the last ReLU of all residual blocks are removed, and we obtain about 0.8% improvement for ResNeXt-29-16x64 on CIFAR-100. The previous state-of-the-art result on CIFAR-100 is DenseNet-BC-190-40, 17.18% error rate. We achieve a better result by applying EraseReLU on a model, which performs worse than DenseNet-BC-190-40. On SVHN, EraseReLU improves the Wide ResNet by about 10% relative accuracy. On CIFAR-10, we can observe general improvements on various kinds of networks. The Wide ResNet-52-10 with EraseReLU does not improve the performance. Because we may close to the lower bound for the CIFAR-10 dataset.

4.3 Experiments on ImageNet

We evaluate the performance of EraseReLU in terms of different architectures on the ImageNet classification task, and compare with the original models. On the subsets of ImageNet, we apply EraseReLU on ResNet-50, ResNet-101 and ResNeXt-152-32x4. We adopt the training code provided by PyTorch [4] and train each network with 120 epochs with the learning rate initialized by 0.1. The learning rate is divided by 10 every 30 epochs. For ResNet-50 and ResNet-101, we simply remove the last ReLU activation layer in each residual block. For ResNeXt-152 (32x4), we only apply EraseReLU on one residual block in each two neighboring residual blocks.

[4] https://github.com/pytorch/examples/blob/master/imagenet/main.py

Sample Ratio		10%	20%	30%
ResNet-50	top-1	47.60	36.76	31.11
	top-5	23.68	15.65	11.65
ResNet-50 with ER	top-1	46.18	56.42	32.58
	top-5	23.20	14.99	12.04

Table 2: Top-1 and top-5 error rate on ImageNet validation subsets using different number of training images. 10% represents we use 10% ImageNet data, *i.e.*, ImageNet-10%. 20% and 30% indicates ImageNet-20% and ImageNet-30%. We compare the results of the original ResNet-50 and the ResNet-50 with EraseReLU, referred as ER.

	top-1 error(%)	top-5 error(%)
ResNet-152	22.2	6.2
ResNet-152 (ER)	21.6	5.8
ResNet-200 †	21.8	6.1
Pre-act ResNet-200	21.7	5.8
ResNet-200 (ER)	**21.4**	5.8

Table 3: The comparison of top-1 and top-5 error rate between various of residual models with and without EraseReLU. They are evaluated on the ImageNet-1K validation set. We only use single-crop for testing with 224×224 image size. "ER" indicates the model using our EraseReLU. † indicates results run by ourselves.

Fig. 5 illustrates the results of ResNet-101 and ResNeXt-152 (32x4). ResNet-101 with EraseReLU outperforms the original model by about 3% (top-1 accuracy). EraseReLU not only obtains a faster convergence rate, but also achieves a higher classification accuracy on the validation set. For ResNeXt-152 (32x4), EraseReLU obtains a similar result compared to the original model. This implies that EraseReLU may also work as an regularizer that benefits the model's generalization performance, especially when data is scarce.

Table 2 show the comparison of ResNet-50 with and without EraseReLU. EraseReLU improves the performance of ResNet-50 on ImageNet-10% about 1.4% absolute top-1 accuracy. When the number of training/validation images increase to ImageNet-20%, the performance improvement goes down to 0.24%. On ImageNet-30%, the top-1 accuracy of ResNet-50 with EraseReLU is lower than ResNet-50. Because as the data scale increase, it can untilize more capacity of ResNet-50, where reducing the nonlinear can harm the performance. For a relatively small dataset, *i.e.* ImageNet-10%, EraseReLU can regularize the model leading a better performance.

On the ImageNet-1k, we apply EraseReLU on two models, ResNet-152 and ResNet-200. To ensure a fair comparison, we adopt the public Torch implementation for ResNet [5] and only replace the model definition by the model applying our EraseReLU. Therefore, all other factors are eliminated, such as data pre-process and data argumentations. We keep all the experiment settings the same as those used for ResNet. For both ResNet-152 and ResNet-200, we remove one last ReLU layer for every two residual blocks. EraseReLU improves the ResNet-152 on ImageNet dataset about 0.6% top-1 accuracy. For ResNet-200, ours achieves 21.4% top-1 accuracy outperforming than the original one about 0.4%. The ResNet-200 with EraseReLU also outperforms than the Pre-act ResNet-200, which is an improved version of ResNet.

5 Conclusion

In this paper, we investigate the effect of erasing ReLUs of certain layers in deep CNN architectures. We find that this seemingly trivial operation leads to a non-negligible improvement of classification performance because its effectiveness on easing the optimization and regularizing the training of deep neural networks. We name this approach "EraseReLU". Base on this approach, we improve the classification performance beyond various kinds of CNN architectures, such as ResNet, Pre-act ResNet, Wide ResNet and ResNeXt. Most of them achieve a much higher performance while having the same computation cost compared to the original models. EraseReLU obtains more than

[5] `https://github.com/facebook/fb.resnet.torch`

2% absolute accuracy improvement on CIFAR-100 compared with the original Pre-act ResNet-1001. It also leads to about 3% absolute accuracy improvement on the 10% subset of ImageNet-1k compared with the ResNet-101, and 0.6% accuracy improvement on the ImageNet-1k compared with ResNet-152.

In future, we will try to give a theoretical explanation about the characteristics of EraseReLU, and apply it to various kinds of computer vision tasks.

REFERENCES

Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *CVPR*, 2017.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. Detecting and recognizing human-object intaractions. *arXiv preprint arXiv:1704.07333*, 2017.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

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

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *ICCV*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *NIPS*, 2015.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.