Learning Strict Identity Mappings in Deep Residual Networks

Xin Yu[1] Zhiding Yu[2] Srikumar Ramalingam[1]
[1] University of Utah [2] NVIDIA
{xiny,srikumar}@cs.utah.com, zhidingy@nvidia.com

Abstract

A family of super deep networks, referred to as residual networks or ResNet [14], achieved record-beating performance in various visual tasks such as image recognition, object detection, and semantic segmentation. The ability to train very deep networks naturally pushed the researchers to use enormous resources to achieve the best performance. Consequently, in many applications super deep residual networks were employed for just a marginal improvement in performance. In this paper, we propose ϵ-ResNet that allows us to automatically discard redundant layers, which produces responses that are smaller than a threshold ϵ, without any loss in performance. The ϵ-ResNet architecture can be achieved using a few additional rectified linear units in the original ResNet. Our method does not use any additional variables nor numerous trials like other hyper-parameter optimization techniques. The layer selection is achieved using a single training process and the evaluation is performed on CIFAR-10, CIFAR-100, SVHN, and ImageNet datasets. In some instances, we achieve about 80% reduction in the number of parameters.

1. Introduction

The basic idea behind ϵ-ResNet is shown in Fig. 1 where ϵ-ResNet is trained on the CIFAR100 dataset [22]. In particular, we show a 752-layer network with each residual block having 2 convolution layers and the "pre-activation" setting following [15]. During the training, we automatically identify the layers that can be pruned or discarded without any loss (or with marginal loss) in the performance. We achieve this by modifying the standard residual network with a few additional rectified linear units that automatically discards residual blocks whose responses are below a threshold. In this particular instance shown, we achieve a compression ratio of around 3.2 (original network size / reduced network size).

Recent advances in representation learning have demonstrated the powerful role played by deep residual learning [14]. As a result, ResNet has pushed the boundaries of a wide variety of vision tasks significantly, including but are not limited to general object recognition [14, 48], object detection [34, 7, 35], face recognition [27], segmentation [5, 47, 52] and semantic boundary detection [50]. More recently, He et al. [15] proposed an improved design of residual unit, where identity mappings are constructed by viewing the activation functions as "pre-activation" of the weight layers, in contrast to the conventional "post-activation" manner. This further led to considerably improved performance on very deep network architectures, such as a 1001-layer ResNet.

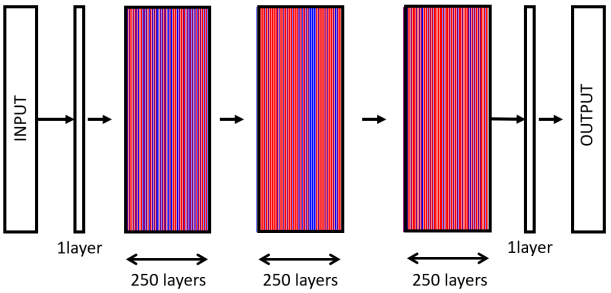
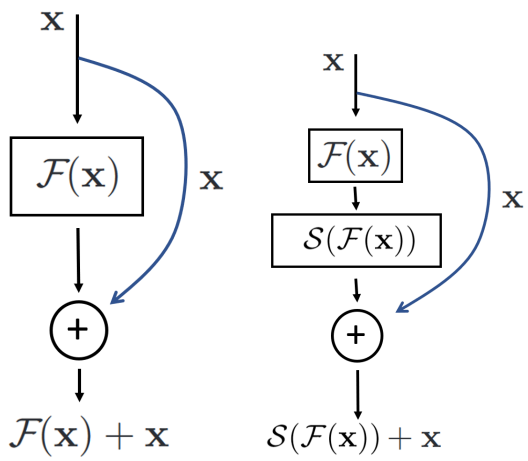


Figure 1. We show a very deep ϵ-residual network with 752 layers for training CIFAR-100 [22]. During training, ϵ-ResNet identifies the layers that can be discarded with marginal or no loss in performance. The red lines indicate the layers that can be pruned, and the blue lines show the layers that need to be used. In this particular instance, we achieve a compression ratio of 3.2 (original number of layers / reduced number of layers). The validation errors of the original and the reduced networks are given by 24.8% and 23.8%, respectively.

The remarkable success of ResNet leads to some obvious questions: What makes it work better than earlier architectures? One advantage with ResNet is its ability to handle vanishing/exploding gradients. However, the success could not be attributed only to this since many prior methods have already handled this with normalized initialization [24, 10, 13]. Another key contributing factor is the depth, which has been proven to be extremely beneficial in model expressiveness [12, 29, 38]. It was observed that training very deep neural networks is not a straightforward



(a) (b)

Figure 2. We show the basic transformation in standard ResNet and ϵ-ResNet (a) The function mapping in standard ResNet is given by $\mathcal{H}(\mathbf{x}) = \mathcal{F}(x) + x$. (b) The function mapping in ϵ-ResNet is given by $\mathcal{H}(\mathbf{x}) = \mathcal{S}(\mathcal{F}(x)) + x$. If all the residual responses in $\mathcal{F}(x)$ is less than a threshold ϵ, then $\mathcal{S}(\mathcal{F}(x)) = \mathbf{0}$. If all the responses are not small, then we do the same mapping $\mathcal{S}(\mathcal{F}(x)) = \mathcal{F}(x)$ as the standard network.

task as we encounter the "under-fitting" problem, where the training error keeps increasing with the depth of the network. This is in stark contrast of the natural expectation of "over-fitting" that typically happens when we use too many parameters. To illustrate this further, let us use consider Fig. 2(a). Let $\mathcal{H}(x)$ be the desired underlying mapping, and we cast it as $\mathcal{F}(x) + x$. Here $\mathcal{F}(x) = \mathcal{H}(x) - x$ is the residual mapping. In [14], the following explanation is provided for the under-fitting problem:

> *"We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers."*

While this makes sense, in reality one seldom observes the residuals going to perfect zeros or even negligible values in experiments with very deep networks. In this paper, we propose a technique that promotes zero residuals, which in other words achieves identity mapping in a strict sense.

A different interpretation for residual networks was given in [44], where ResNet was treated as an ensemble of many shallow networks. In particular, they show several interesting experiments to demonstrate the role of depth and layers in VGGNet and ResNet. In particular, they show the deletion of one layer in VGGNet can lead to 80% increase in error, while one barely notices the difference in ResNet.

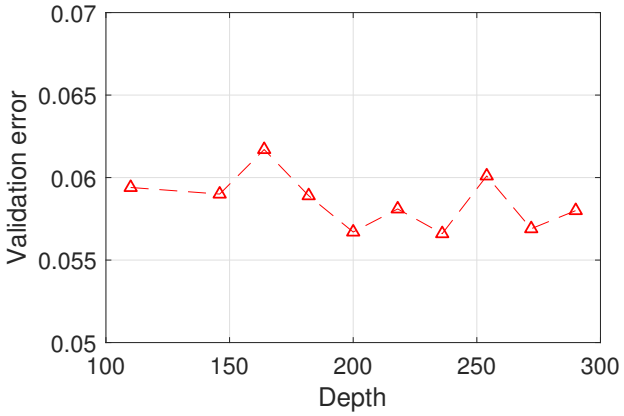


Figure 3. validation errors of full pre-activation ResNet with different number of layers in CIFAR10 dataset [22].

This shows that depth alone may not be the single key factor in the success of ResNet. Their hypothesis is that the multiple short and sometimes redundant paths play an important role in the performance. In order to study the role of depth in residual networks, we performed a simple experiment with CIFAR10 dataset [22]. We trained a family of deep residual networks with monotonically increasing number of layers from 100 to 300 as shown in Fig. 3. As we observe, the error on the validation set is not monotonically decreasing. For example, the validation errors of 254-layer and 200-layer networks are given by 5.96% and 5.66%. Although there is a overall decrease in the error rate as we increase the number of layers, the behaviour is not strictly monotonic. Therefore, we would like to ask the following questions:

> *By training a residual network $\mathcal{N}$ with n layers, can we find a reduced network $\mathcal{N}_R$ with $m \ll n$ layers without significant performance loss?*

In this paper we propose ϵ-ResNet, a variant of standard residual networks, that promotes strict identity mapping. We illustrate the basic idea behind ϵ-ResNet in Fig. 2(b). We model the desired unknown mapping as $\mathcal{H}(\mathbf{x}) = \mathcal{S}(\mathcal{F}(\mathbf{x})) + \mathbf{x}$. When all the responses in the original residual block $\mathcal{F}(\mathbf{x})$ is below a threshold ϵ, then we force $\mathcal{S}(\mathcal{F}(\mathbf{x})) = \mathbf{0}$. If any single response is not small, then we use the original mapping $\mathcal{S}(\mathcal{F}(\mathbf{x})) = \mathcal{F}(\mathbf{x})$ from the standard residual network. In this paper, we will show that when a residual block produces zero responses using our proposed variant, then the weights in the CNN filters of the corresponding residual block will be pushed to zeros by the training loss function that consists of cross-entropy term and $L2$ norm of the weight parameters with momentum optimization [20]. Consequently, during the prediction or test time, we can safely remove these layers and build a reduced network. A direct benefit of such framework

is model compression and faster inference when there is layer redundancy. In our experiments, ϵ-ResNet produces remarkable reduction of the model size with marginal or no loss in performance. Our expectation is that ϵ-ResNet should at least achieve good trade-off between performance and model size/complexity.

2. Related Work

Residual Networks and variants: A significant direction of network design has been focusing on designing skip layer architectures in order to alleviate the vanishing/exploding gradient issues [41]. More recent advances of this direction have led to the family of deep residual networks [14, 48, 42]. He et al. further proposed an improved residual unit with full pre-activation [15], where identity mappings are used as skip connections. Under this framework, it was mathematically shown that the feature of any deeper unit can be represented as the feature of any shallower unit plus the summation of the preceding residual responses. Such characteristic leads to significantly improved performance on ultra deep networks with more than 1000 layers, while previously proposed residual units suffer from over-fitting and degraded performance. However, saturation of performance gain still exists as the layer number increases, meaning that there is room for removing redundancy and achieve better trade-off between performance and model complexity.

Network structure optimization: The problem addressed in this paper can be seen as one of the subproblems of hyper-parameter optimization, where we identify a subnetwork by dropping unnecessary layers without any (or marginal) loss in the performance. We will briefly review the underlying ideas in hyper-parameter optimization. One of the distinctly unsolved problems in deep learning is the ability to automatically choose the right network architecture for solving a particular task. The popular ones (e.g., AlexNet, GoogLeNet, and Residual Networks) have shown record beating performance on various image recognition tasks. Nevertheless, it is a time-consuming process to decide on the hyper-parameters (HPs) of a network such as the number of layers, type of activation functions, learning rate, and loss functions.

Non-practitioners of deep learning might ask: why is it difficult to optimize a few HPs, while we already train millions of network weight parameters? One can view this as a global optimization of a black-box loss function f. The goal is to find the HPs θ_h that minimizes $f(\theta_h, \theta_w, \mathcal{D}_{val})$, such that $\theta_w = \arg\min_{\theta_w} f(\theta_h, \theta_w, \mathcal{D}_{train})$. Here θ_w, $\mathcal{D}_{train}$, and $\mathcal{D}_{val}$ denote the weight parameters, training dataset, and validation dataset, respectively. Some of the HPs such as the depth are discrete, and the topology of the network changes for various depth values. This makes it hard to treat both sets of parameters θ_h and θ_w in the same manner. Existing approaches evaluate the $f(\theta_h, \theta_w, \mathcal{D}_{val})$ for different values of θ_h and identify the optimal θ_h.

Standard approaches include search strategies (manual, grid, and random) and Bayesian techniques. A grid search is a brute-force strategy that evaluates the function for all HP values in a manually specified interval. In a random search, we evaluate on a random subset of parameter values to identify the optimum. Manual search denotes the process of greedily optimizing one parameter at a time, and then moving on to the next one – we memorize and analyze our previous results, and this gives us an advantage over naive grid or random search. Practical considerations in the manual tuning of HPs are provided for efficient training and debugging of large-scale networks [2].

Bayesian methods [40, 4] can automate the process of manual tuning by learning a statistical model of the function that maps the hyper-parameter values to the performance on the validation set. In other words, we can think of Bayesian methods as modeling the conditional probability $p(f|\theta_h)$ where f is the performance on the validation set given HPs θ_h. By studying the performance of different search techniques on the 117 datasets from [9], it was shown that many recent methods are only marginally better than random search [33]. It is highly recommended that all HP optimization methods should be compared with random search baseline [3]. Reinforcement learning [54] and evolutionary algorithms [32] have also been used, but these methods use evaluations for a large number of parameter trials, and it is time-consuming for even small-scale problems. While many of these methods are driven towards finding the optimal network architecture that produces best performance, our work focuses on a subproblem where we identify a subnetwork that produces more-or-less the same results as the original one in a specific case of residual networks.

Many other researchers have looked at model compression using other techniques such as low-rank decomposition [8, 21, 51], quantization [31, 6, 46], architecture design [43, 19, 17], pruning [11, 25, 28], sparse [26, 53, 1, 45] learning, etc. Recently, sparse structure selection has been shown for residual networks using scaling factor variables [18]. These additional scaling variables are trained along with the standard weight parameters using $L1$ regularization terms using stochastic Accelerated Proximal Gradient (APG) method. While [18] and ϵ-ResNet share the same goal of discarding redundant layers, the techniques are entirely different. We do not add any additional variables to the standard residual networks. While [18] uses $L1$ relaxation for solving $L0$ sparsity in the loss function, we promote layer sparsity by redesigning the network architecture that can achieve strict identity mapping.

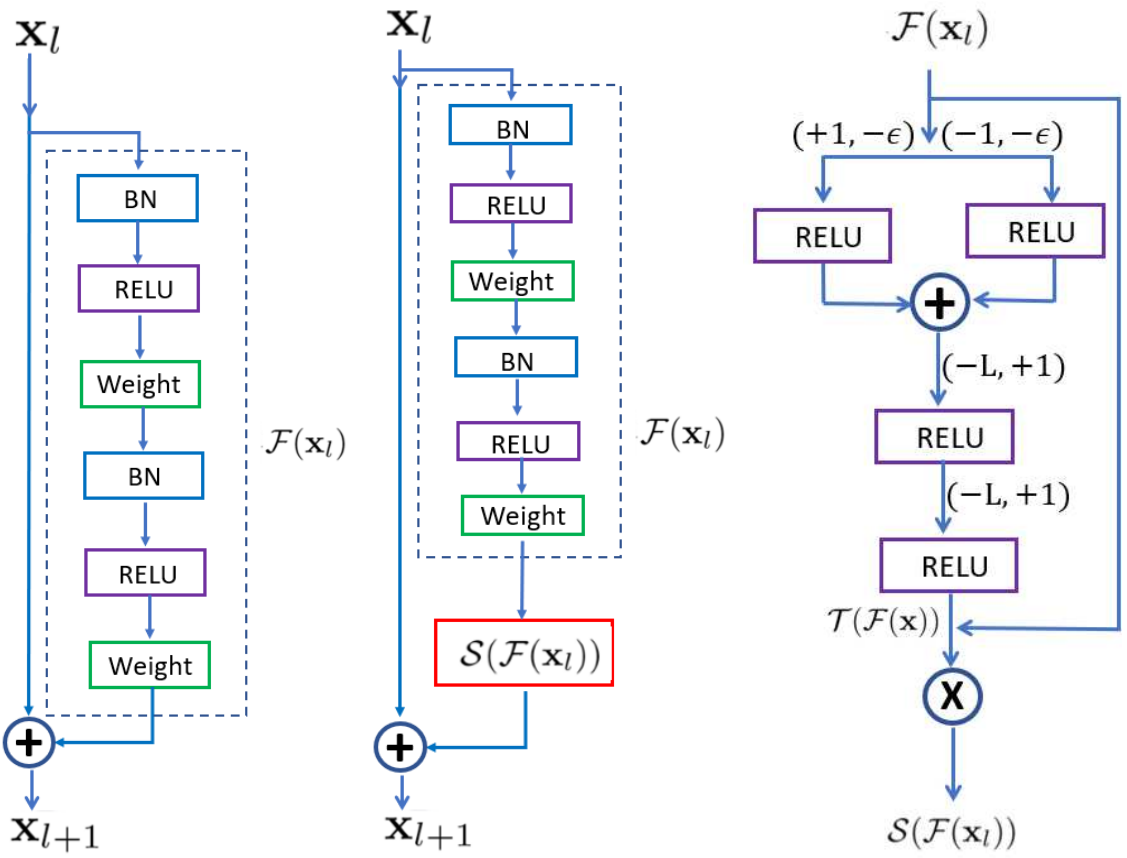


Figure 4. (a) We show the network structure of one of the residual blocks in the standard pre-activation network [15]. (b) We show the network structure of one of the residual blocks in ϵ-ResNet. We have added a sparsity-promoting function $\mathcal{S}()$ to discard the residual if all the individual responses are less than a threshold ϵ. (c) We show the network structure for the sparsity-promoting function $\mathcal{S}()$ using 4 RELU's and one multiplicative gate. The pair (i, j) shown in brackets denote the weights i and bias term j for the associated RELU function. L refers to a very large positive constant and ϵ denotes the constant that we want to use for discarding layers.

3. ϵ-ResNet

Standard ResNet: Residual network consists of a large stack of residual blocks. Each residual block has the architecture shown in Fig. 4(a). Each residual block can be seen as the following mapping:

$$\mathcal{H}(\mathbf{x}) = \mathbf{x} + \mathcal{F}(\mathbf{x}) \tag{1}$$

The residual block consists of pre-activations of weight layers as proposed in the improved version of residual networks [15]. Note that this is in contrast to the earlier version that used post-activations (RELU and Batch normalization BN) of weight layers [14]. In all our experiments, we use the pre-activation residual network as the baseline. Our proposed ϵ-ResNet will also be built on top of this pre-activation residual networks. The entire network is built by stacking multiple residual blocks. It has three groups of residual blocks, and each group has equal numbers of blocks. For example, in the case of a 110-layer network, we have 3 groups each having 18 blocks. Each block has 2 layers, and thus the three groups will have a total of 108 ($3 \times 2 \times 18$) layers. In addition, we have one convolution layer before and one fully connected layer after the three groups of residual blocks, and thereby leading to 110 layers. The dimension of the first group is 16 and is multiplied by two in the later two groups, and the feature-map sizes in the three groups are $32 \times 32, 16 \times 16, 8 \times 8$.

ϵ-ResNet: Fig. 4(b) shows the mapping function in every block in the ϵ-ResNet. The basic idea in ϵ-ResNet is to use a **sparsity-promoting function** $\mathcal{S}(\mathcal{F}(\mathbf{x}))$ that automatically discards the residual $\mathcal{F}(\mathbf{x})$ if all the individual responses are less than a threshold ϵ. Instead using a function mapping $\mathcal{H}(\mathbf{x}) = \mathbf{x} + \mathcal{F}(\mathbf{x})$, we use a function mapping as shown below:

$$\mathcal{H}(\mathbf{x}) = \mathbf{x} + \mathcal{S}(\mathcal{F}(\mathbf{x})) \tag{2}$$

Let us assume that $\mathcal{F}(\mathbf{x})$ is a vector of length n. Let each element of the vector is denoted by $\mathcal{F}(\mathbf{x})_i$ where $i \in \{1, \ldots, n\}$. The sparsity-promoting function is defined below:

$$\mathcal{S}(\mathcal{F}(\mathbf{x})) = \begin{cases} 0 & \text{if } |\mathcal{F}(\mathbf{x})_i| < \epsilon, \forall i \in \{1, \ldots, n\} \\ \mathcal{F}(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{3}$$

Proposed Structure: Fig. 4(c) shows our proposed network structure to implement the sparsity promoting function $\mathcal{S}(\mathcal{F}(\mathbf{x}))$. The network consists of 4 RELU layers and one multiplicative gating function $(\times)$, as utilized in [41].

Let us study the behaviour of this network. First, let us consider the case when all the response elements satisfy the condition $|\mathcal{F}(\mathbf{x})_i| < \epsilon$. In this case, the output from the summation $(+)$ will be zero. A zero input to the third RELU will lead to an output of 1. An output of 1 from the third RELU will lead to an output of 0 from the fourth RELU. Finally, we will have $\mathcal{T}(\mathcal{F}(\mathbf{x})) = 0$, and thus we have the following:

$$\mathcal{S}(\mathcal{F}(\mathbf{x})) = \mathcal{T}(\mathcal{F}(\mathbf{x})) \times \mathcal{F}(\mathbf{x}) = 0 \tag{4}$$

Now let us consider the second scenario where at least one of the response elements satisfies either $\mathcal{F}(\mathbf{x}) > \epsilon$ or $\mathcal{F}(\mathbf{x}) < -\epsilon$. In this case we will have a non-zero positive output from the first summation $(+)$. Non-zero positive output from the summation would lead to 0 output after the third RELU, and eventually output $\mathcal{T}(\mathcal{F}(\mathbf{x})) = 1$ from the final RELU. Thus we have:

$$\mathcal{S}(\mathcal{F}(\mathbf{x})) = \mathcal{T}(\mathcal{F}(\mathbf{x})) \times \mathcal{F}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) \tag{5}$$

Loss function: We use the same loss function that was used in ResNet [15] involving the cross-entropy term and L2 regularization on the weights. Note that we also use a side-supervision in addition to the loss function at the last output layer. Let us denote the additional function used for side-supervision as the side loss. If there are N residual blocks, the output of [N/2]-th block is used in the computation of the side loss. To get side loss, we follow the same architecture as the standard loss function. We apply a fully-connected layer before the soft-max layer and then forward their output to the cross entropy function. The output dimension of the fully-connected layer is the number of classes in the dataset. Finally, side loss is used in the overall cost function with a coefficient of 0.1. Side supervision can be seen as a strategy to shorten the path of back propagation for the first half of layers during training. Note that side supervision is not involved in prediction.

Weight Collapse: When a residual block produces negligible responses, the sparsity promoting function will start producing 0 outputs. As a result, the weights in this block will stop contributing to the cross-entropy term. Consequently the gradients will be only based on the regularization term, and thus the weights in the associated residual block will move to 0's. Note that the weights don't go to zeros instantly, and the number of iterations necessary for reaching zeros depends on the learning rate and momentum parameters. In Fig. 5, we show the weight collapse for one of the layers in a residual block that starts to model strict identity mapping.

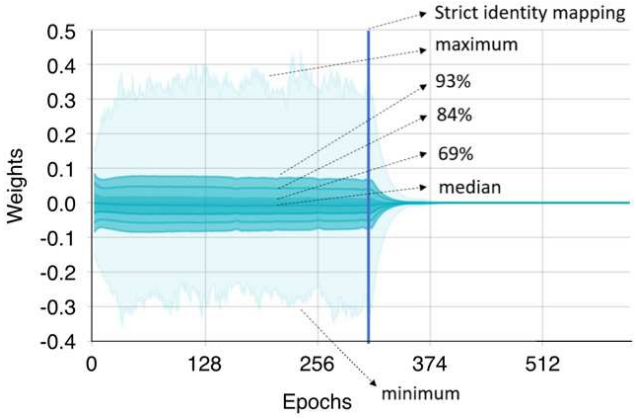


Figure 5. We show the histogram of all weights in one of the layers of a residual block that achieves strict identity. The x-axis shows the index for the iterations. The y-axis shows the value of the weights. The different curves show the maximum, minimum, 93rd percentile, etc. As we observe, the weights collapse to zeros once a residual block is identified as unnecessary by ourϵ-ResNet algorithm.

4. Experiments

We evaluate ϵ-ResNet on four standard datasets: CIFAR-10 and CIFAR-100 [22], SVHN [30], and ImageNet 2012 dataset [36]. We used standard ResNet as the baseline for comparison.

Datasets: The CIFAR-10 dataset [22] consists of $60,000$ 32×32 RGB images with 10 classes, where each class has $6,000$ images. The CIFAR-100 dataset [22] also consists of $60,000$ 32×32 RGB images with 100 classes, where each class has 600 images. We use $50,000$ training images and $10,000$ test images in both the datasets. We follow the standard data augmentation that allows for small translations and horizontal flips. We first pad 4 pixels on all sides, then we randomly crop 32×32 images from the

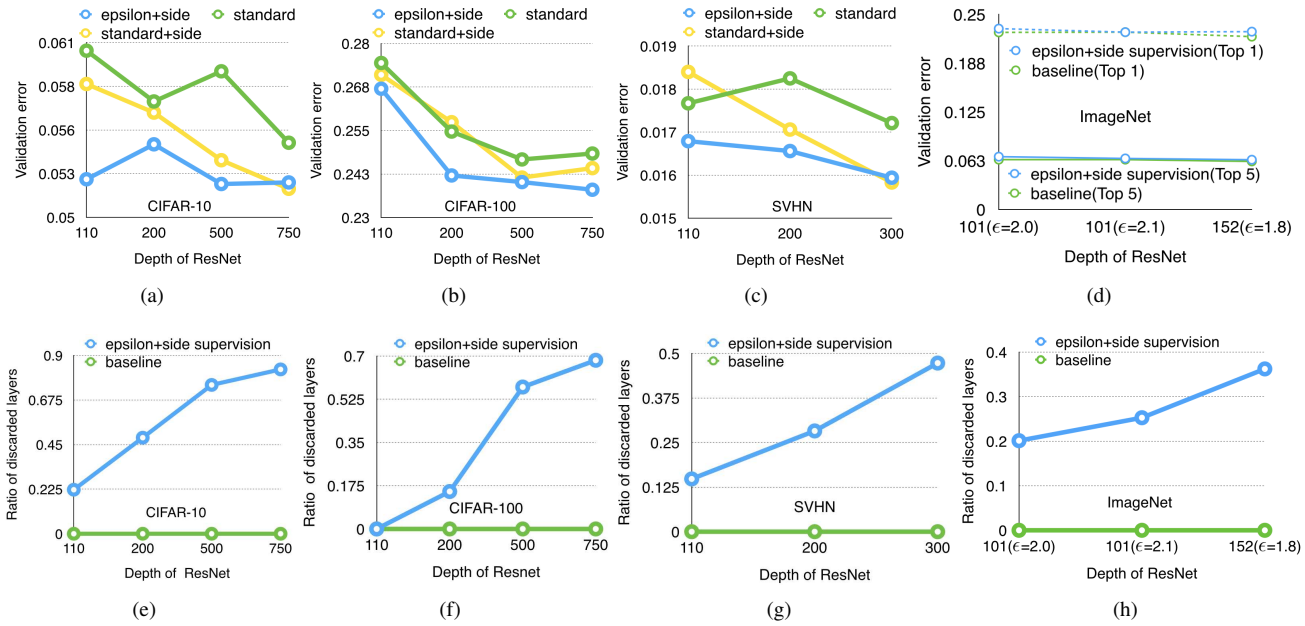


Figure 6. (a), (b), (c), and (d) show the validation error of ResNet and ϵ-ResNet with different number of layers on CIFAR-10, CIFAR-100, SVHN, and ImageNet, respectively. (e), (f), (g), and (h) show the ratio of discarded layers of ϵ-ResNet with a different number of layers on CIFAR-10, CIFAR-100, SVHN, and ImageNet, respectively. The validation error of ResNet-152 baseline is borrowed from https://github.com/facebook/fb.resnet.torch.

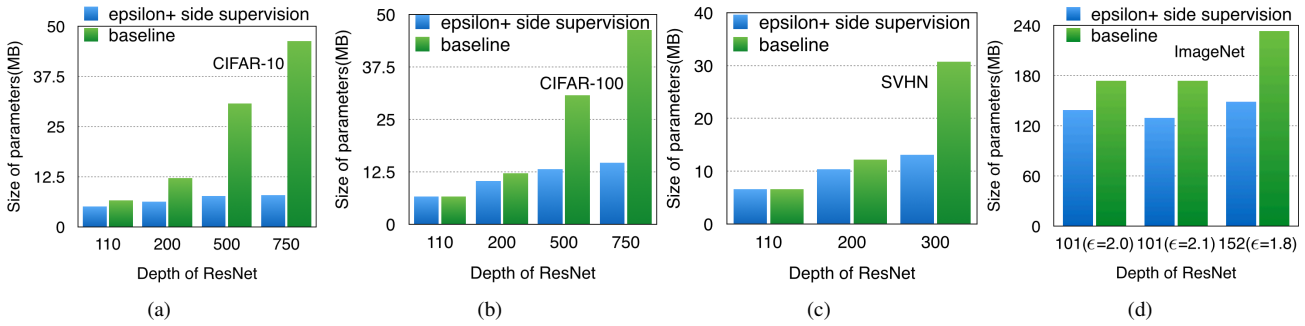


Figure 7. (a), (b), and (c) show the memory consumption for parameters used in the standard ResNet and the reduced one for CIFAR-10, CIFAR-100, and ImageNet experiments, respectively. As we can see, we achieve a significant compression in the case of networks with a large number of layers.

padded image, and finally we do a horizontal flip.

In the Street View House Numbers (SVHN) [30] dataset, the models are trained on $604,388$ RGB images and evaluated on $26,032$ RGB images of size 32×32. For data augmentation, we apply the same augmentation method as on the CIFAR datasets. Before random cropping, we apply the following additional transformations to augment the data: For every training image, we perform scaling and transformation with one random factor on all its pixels. In particular, we add a random number in $[-10, 10]$ for all pixels on one image, then for each pixel randomly multiply the residual between RGB channel values and their channel mean with a scale between $[0.8, 1.2]$, add the scaled residuals to the original mean. The randomly scaled and transformed values are then truncated to the range of $[0, 255]$ before output.

The ImageNet 2012 classification dataset [36] has $1,000$ classes and contains $1,281,167$ training images and $50,000$ validation images. We report both top-1 and top-5 validation error rates on it. We follow the practice in [14, 16, 43, 23] to conduct data augmentation on ImageNet. A 224×224 crop is randomly sampled from an image by

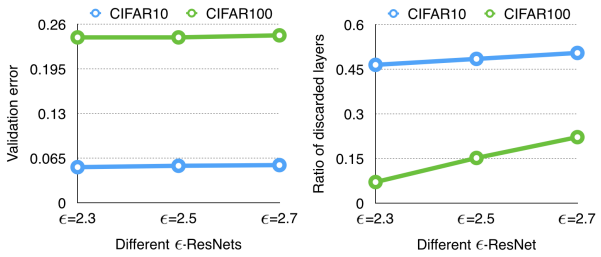


Figure 8. Sensitivity to the choice of ϵ parameter values: We studied the ratio of discarded layers and validation errors for different ϵ parameter values in the case of CIFAR-10 and CIFAR-100. We observed that ϵ parameter can be seen as a tuning parameter to achieve the tradeoff between compression and accuracy.

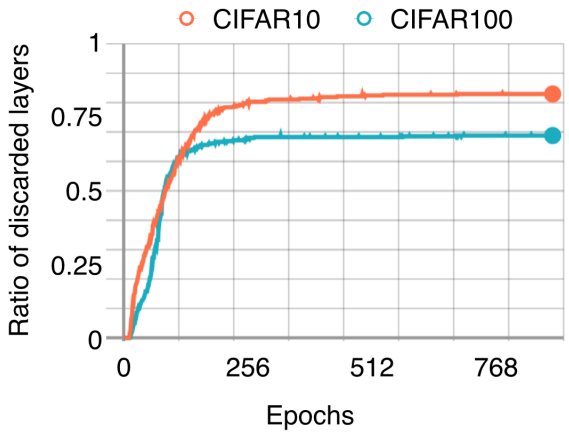


Figure 9. We study the ratio of discarded layers with respect to the number of epochs. Results of a 752-layer ϵ-ResNet on CIFAR-10 and CIFAR-100.

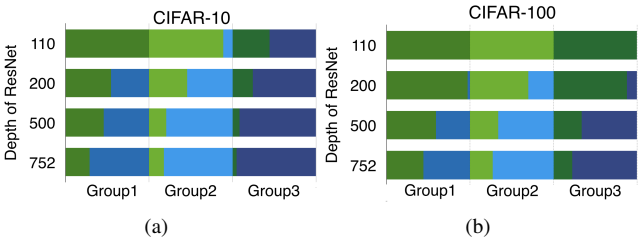


Figure 10. Group-wise proportions of layers that are not discarded versus the discarded ones on ResNets with different depths. All network structures in the experiment contain three groups, where each group contains layers at the same resolution. For each group, green bar represents the proportion of layers not discarded, while blue bar represents the proportion of discarded layers. (a) and (b) show the results on CIFAR-10 and CIFAR-100, respectively.

following the scale and aspect ratio augmentation from [43], instead of scale augmentation used in the ResNet [14] paper because the former gives a better validation error. We then apply the following transformations in a random order: horizontal flip, the standard AlexNet-style color augmentation [23] and photometric distortions [16].

ϵ Parameter: We can think of ϵ as a hyper-parameter that allows us to find a tradeoff between accuracy and network size. In practice, identifying a good ϵ parameter is not difficult due to the following reasons: (1) The ϵ values are generally concentrated within a small range to produce good results and compression, as shown in the experiments. (2) ϵ satisfies nice monotonicity property (larger value leads to higher compression). (3) In all the experiments (CIFAR-10, CIFAR-100, SVHN, and Imagenet), we were able to quickly find ϵ parameters (generally in the range 1.5-3 with just 1 or 2 attempts). For example, the experiments for CIFAR-10 and CIFAR-100 used an ϵ of 2.5. For all experiments on SVHN, we use an ϵ of 1.5. We use ϵ in the range (1.8, 2.1) on ImageNet. Fig. 8 shows that in a reasonable range, greater ϵ brings larger compression ratio.

Number of layers: We experimented with networks of depths 110, 200, 500, and 752-layers on CIFAR-10 and CIFAR-100. These four networks have 54, 99, 249, 375 residual blocks (consisting of two 3×3 convolutional layers each) respectively. Considering the size of SVHN is 10 times larger than CIFAR-10, we test SVHN with networks of depths 100, 200, and 302 layers. Finally, we evaluated the models of depths 101 and 152 on ImageNet.

Adaptive learning rate: Our learning rate scheduling closely follows the standard ResNet implementation [14]. On CIFAR dataset, we start with a learning rate of 0.1 and decrease it by a factor of 10 at epochs 82 and 123. On SVHN dataset, we begin with 0.1 and decrease it by a factor of 10 at epochs 20, 28 and 50. When the network starts to lose layers, we will stop using the standard learning rate policy and start using adaptive learning rate policy. According to our adaptive learning rate policy, every time we lose a residual block, we reset the learning rate to the initial value of the standard setting and decrease it at a rate that is twice as fast as the standard policy. In other words, we decrease it by a factor of 10 after 41 and 61 epochs for CIFAR datasets. For SVHN, we will start decreasing by a factor 10 after 10, 14, and 25 epochs. Such adaptive learning rate policies have been used before [39, 37].

Adaptive learning rate policy was not necessary for ImageNet, where we just start with a learning rate of 0.1 and decrease it by a factor of 10 at epochs 30, 60, 85, and 95.

Training: We implement ϵ-ResNet using Tensorflow on TITAN XP and GeForce GTX 970 graphic cards. Our code is built upon the tensorpack, an implementation of ResNet can be found here. We followed the standard Gaussian initialization of 0 mean and 0.01 std for weights, and constant

initialization of 0s for biases.

Following the practice in [14], we use standard data augmentation methods and train the network using SGD with a mini-batch size of 128 on CIFAR and SVHN datasets and a mini-batch size 256 on ImageNet for each GPU. On CIFAR datasets the baseline models are trained for up to 200 epochs ($78,000$ iterations), while ϵ-ResNets requires $1,000$ epochs to train since it uses adaptive learning rate. For SVHN, the baseline and ϵ-Resnet used 80 and 150 epochs, respectively. All the models on ImageNet are trained for 110 epochs. We use a weight decay of 0.0002 and a momentum of 0.9 for all the experiments.

Side-supervision: Side-supervision is a strategy to impose additional losses at intermediate hidden layers in addition to the loss on the top [49]. In ϵ-ResNet, we apply one additional loss at the middle of the network with a coefficient of 0.1. We observed that side supervision allows the weights to generally decrease as we move from the input layer to the output layer. As shown in Fig. 1, we reject more layers near the output layer as shown by the red lines.

In Fig. 6, we also show the results for the standard Resnet improves with side-supervision. However, ϵ-ResNet still achieves similar performance, along with providing the additional benefit of significant compression.

Validation Errors: Fig. 6 report the comparison in validation errors between standard ResNet and ϵ-ResNet. ϵ-ResNet again achieves a significant reduction in model size while maintaining good prediction performance on CIFAR-10, CIFAR-100, and SVHN. On these three datasets, we did not find any degradation in the performance even after discarding a significant number of layers. We also evaluated the performance of ϵ-ResNet with 101 and 152 layers on ImageNet. With a marginal loss in performance, ϵ-ResNet discarded 20.12%, 25.60%, and 36.23% of layers for different ϵ parameters and depths.

Memory consumption: The memory footprint reduction for CIFAR-10, CIFAR-100, SVHN, and ImageNet are shown in Fig. 7, respectively.

Layer Selection Fig. 9 shows that the proportion of strict identity mapping increases with the training iterations and gradually saturates, finally stabilizing at a ratio where the performance is similar to the original. When looking into the final learned structure, Fig. 10 shows that ϵ-ResNet is prone to discarding more layers closer to the output.

5. Discussion

We propose ϵ-ResNet, a variant of standard residual networks [15], that automatically identifies and discards redundant layers with marginal or no loss in performance. We achieve this using a novel architecture that enables strict identity mappings when all the individual responses from a layer are smaller than a threshold ϵ. We tested the proposed architecture on four datasets: CIFAR-10, CIFAR-100, SVHN, and ImageNet. We plan to explore two avenues in future. First, we will focus on developing an algorithm that can automatically identify a good ϵ value that produces maximum compression with marginal or no loss in performance. Second, we will extend the proposed method to architectures beyond ResNet.

Acknowledgments

We thank the reviewers and area chairs for valuable feedback. S. Ramalingam would like to thank Mitsubishi Electric Research Laboratories (MERL) for partial support. We thank Tolga Tazdizen and Vivek Srikumar for GPU resources.

References

[1] J. M. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In *NIPS*, 2016. 3

[2] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012. 3

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 2012. 3

[4] J. S. Bergstra and et al. Algorithms for hyper-parameter optimization. In *NIPS*, 2011. 3

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. PAMI*, 2017. 1

[6] M. Courbariaux and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained +1 or -1. In *NIPS*, 2016. 3

[7] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1

[8] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 3

[9] M. Feurer and et al. Efficient and robust automated machine learning. In *NIPS*, 2015. 3

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 1

[11] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 3

[12] J. Hastad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1991. 1

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 1

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 4, 6, 7, 8

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1, 3, 4, 5, 8

[16] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *CoRR*, abs/1312.5402, 2013. 6, 7

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[18] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. *CoRR*, abs/1707.01213, 2017. 3

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 3

[20] S. Ilya, M. James, D. George, and H. Geoffrey. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 2013. 2

[21] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *BMVC*, 2014. 3

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 1, 2, 5

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 6, 7

[24] Y. Lecun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. *In Neural Networks: Tricks of the Trade*, 1998. 1

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 3

[26] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *CVPR*, 2015. 3

[27] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 1

[28] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017. 3

[29] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014. 1

[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS*, 2011. 5, 6

[31] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 3

[32] E. Real and et al. Large-scale evolution of image classifiers. In *ICML*, 2017. 3

[33] B. Recht. The news on hypertuning. http://www.argmin.net/2016/06/20/hypertuning/, 2016. 3

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. PAMI*, 39(6):1137–1149, 2017. 1

[35] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *IEEE Trans. PAMI*, 39(7):1476–1481, 2017. 1

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F. Li. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 5, 6

[37] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *ICML*, 2013. 7

[38] T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. *CoRR*, abs/1711.02114, 2017. 1

[39] L. N. Smith. Cyclical learning rates for training neural networks. In *WACV*, 2017. 7

[40] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012. 3

[41] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In *ICML*, 2015. 3, 5

[42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 3

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3, 6, 7

[44] A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016. 2

[45] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016. 3

[46] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, 2016. 3

[47] Z. Wu, C. Shen, and A. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016. 1

[48] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 3

[49] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 8

[50] Z. Yu, C. Feng, M. Liu, and S. Ramalingam. Casenet: Deep category-aware semantic edge detection. In *CVPR*, 2017. 1

[51] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *CVPR*, 2015. 3

[52] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1

[53] H. Zhou, J. M. Alvarez, and F. Porikli. Less is more: Towards compact cnns. In *ECCV*, 2016. 3

[54] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *https://arxiv.org/abs/1611.01578*, 2017. 3