Identity Mappings in Deep Residual Networks

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

Microsoft Research

Abstract Deep residual networks [1] have emerged as a family of extremely deep architectures showing compelling accuracy and nice convergence behaviors. In this paper, we analyze the propagation formulations behind the residual building blocks, which suggest that the forward and backward signals can be directly propagated from one block to any other block, when using identity mappings as the skip connections and after-addition activation. A series of ablation experiments support the importance of these identity mappings. This motivates us to propose a new residual unit, which makes training easier and improves generalization. We report improved results using a 1001-layer ResNet on CIFAR-10 (4.62% error) and CIFAR-100, and a 200-layer ResNet on ImageNet. Code is available at: https://github.com/KaimingHe/resnet-1k-layers.

1 Introduction

Deep residual networks (ResNets) [1] consist of many stacked "Residual Units". Each unit (Fig. 1 (a)) can be expressed in a general form:

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l),$$
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l),$$

where $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$ are input and output of the l-th unit, and $\mathcal{F}$ is a residual function. In [1], $h(\mathbf{x}_l) = \mathbf{x}_l$ is an identity mapping and f is a ReLU [2] function.

ResNets that are over 100-layer deep have shown state-of-the-art accuracy for several challenging recognition tasks on ImageNet [3] and MS COCO [4] competitions. The central idea of ResNets is to learn the additive residual function $\mathcal{F}$ with respect to $h(\mathbf{x}_l)$, with a key choice of using an identity mapping $h(\mathbf{x}_l) = \mathbf{x}_l$. This is realized by attaching an identity skip connection ("shortcut").

In this paper, we analyze deep residual networks by focusing on creating a "direct" path for propagating information — not only within a residual unit, but through the entire network. Our derivations reveal that *if both* $h(\mathbf{x}_l)$ *and* $f(\mathbf{y}_l)$ *are identity mappings*, the signal could be *directly* propagated from one unit to any other units, in both forward and backward passes. Our experiments empirically show that training in general becomes easier when the architecture is closer to the above two conditions.

To understand the role of skip connections, we analyze and compare various types of $h(\mathbf{x}_l)$. We find that the identity mapping $h(\mathbf{x}_l) = \mathbf{x}_l$ chosen in [1]

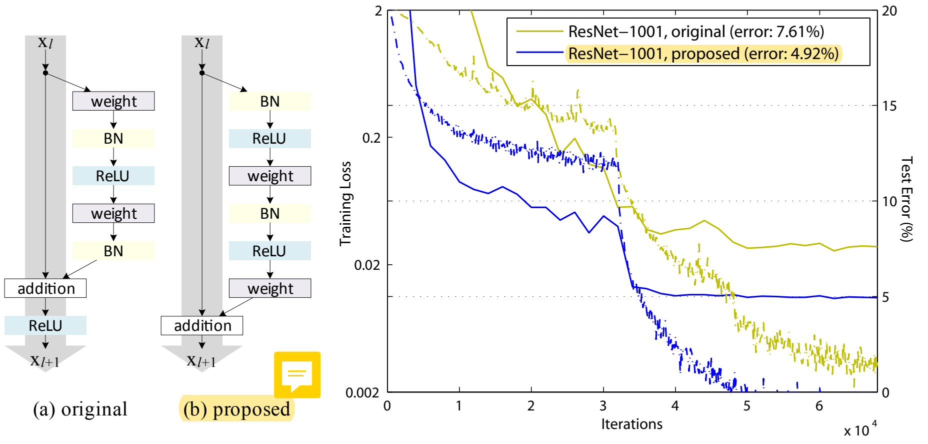

Figure 1. Left: (a) original Residual Unit in [1]; (b) proposed Residual Unit. The grey arrows indicate the easiest paths for the information to propagate, corresponding to the additive term "$\mathbf{x}_l$" in Eqn.(4) (forward propagation) and the additive term "1" in Eqn.(5) (backward propagation). **Right**: training curves on CIFAR-10 of **1001-layer** ResNets. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left). The proposed unit makes ResNet-1001 easier to train.

achieves the fastest error reduction and lowest training loss among all variants we investigated, whereas skip connections of scaling, gating [5,6,7], and 1×1 convolutions all lead to higher training loss and error. These experiments suggest that keeping a "clean" information path (indicated by the grey arrows in Fig. 1, 2, and 4) is helpful for easing optimization.

To construct an identity mapping $f(\mathbf{y}_l) = \mathbf{y}_l$, we view the activation functions (ReLU and BN [8]) as "*pre-activation*" of the weight layers, in contrast to conventional wisdom of "post-activation". This point of view leads to a new residual unit design, shown in (Fig. 1(b)). Based on this unit, we present competitive results on CIFAR-10/100 with a 1001-layer ResNet, which is much easier to train and generalizes better than the original ResNet in [1]. We further report improved results on ImageNet using a 200-layer ResNet, for which the counterpart of [1] starts to overfit. These results suggest that there is much room to exploit the dimension of *network depth*, a key to the success of modern deep learning.

2 Analysis of Deep Residual Networks

The ResNets developed in [1] are *modularized* architectures that stack building blocks of the same connecting shape. In this paper we call these blocks "*Residual*

Units". The original Residual Unit in [1] performs the following computation:

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \tag{1}$$

$$\mathbf{x}_{l+1} = f(\mathbf{y}_l). \tag{2}$$

Here $\mathbf{x}_l$ is the input feature to the l-th Residual Unit. $\mathcal{W}_l = \{\mathrm{W}_{l,k}|_{1 \le k \le K}\}$ is a set of weights (and biases) associated with the l-th Residual Unit, and K is the number of layers in a Residual Unit (K is 2 or 3 in [1]). $\mathcal{F}$ denotes the residual function, *e.g.*, a stack of two 3×3 convolutional layers in [1]. The function f is the operation after element-wise addition, and in [1] f is ReLU. The function h is set as an identity mapping: $h(\mathbf{x}_l) = \mathbf{x}_l$.[1]

If f is also an identity mapping: $\mathbf{x}_{l+1} \equiv \mathbf{y}_l$, we can put Eqn.(2) into Eqn.(1) and obtain:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l). \tag{3}$$

Recursively ($\mathbf{x}_{l+2} = \mathbf{x}_{l+1} + \mathcal{F}(\mathbf{x}_{l+1}, \mathcal{W}_{l+1}) = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) + \mathcal{F}(\mathbf{x}_{l+1}, \mathcal{W}_{l+1})$, etc.) we will have:

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i), \tag{4}$$

for *any deeper unit* L and *any shallower unit* l. Eqn.(4) exhibits some nice properties. **(i)** The feature $\mathbf{x}_L$ of any deeper unit L can be represented as the feature $\mathbf{x}_l$ of any shallower unit l plus a residual function in a form of $\sum_{i=l}^{L-1} \mathcal{F}$, indicating that the model is in a *residual* fashion between any units L and l. **(ii)** The feature $\mathbf{x}_L = \mathbf{x}_0 + \sum_{i=0}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$, of any deep unit L, is the *summation* of the outputs of all preceding residual functions (plus $\mathbf{x}_0$). This is in contrast to a "plain network" where a feature $\mathbf{x}_L$ is a series of matrix-vector *products*, say, $\prod_{i=0}^{L-1} W_i \mathbf{x}_0$ (ignoring BN and ReLU).

Eqn.(4) also leads to nice backward propagation properties. Denoting the loss function as $\mathcal{E}$, from the chain rule of backpropagation [9] we have:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left(1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \right). \tag{5}$$

Eqn.(5) indicates that the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l}$ can be decomposed into two additive terms: a term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ that propagates information directly without concerning any weight layers, and another term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left(\frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathcal{F} \right)$ that propagates through the weight layers. The additive term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ ensures that information is directly propagated back to *any shallower unit* l. Eqn.(5) also suggests that it

[1] It is noteworthy that there are Residual Units for increasing dimensions and reducing feature map sizes [1] in which h is not identity. In this case the following derivations do not hold strictly. But as there are only a very few such units (two on CIFAR and three on ImageNet, depending on image sizes [1]), we expect that they do not have the exponential impact as we present in Sec. 3. One may also think of our derivations as applied to all Residual Units within the same feature map size.

is unlikely for the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l}$ to be canceled out for a mini-batch, because in general the term $\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathcal{F}$ cannot be always -1 for all samples in a mini-batch. This implies that the gradient of a layer does not vanish even when the weights are arbitrarily small.

Discussions

Eqn.(4) and Eqn.(5) suggest that the signal can be directly propagated from any unit to another, both forward and backward. The foundation of Eqn.(4) is two identity mappings: (i) the identity skip connection $h(\mathbf{x}_l) = \mathbf{x}_l$, and (ii) the condition that f is an identity mapping.

These directly propagated information flows are represented by the grey arrows in Fig. 1, 2, and 4. And the above two conditions are true when these grey arrows cover no operations (expect addition) and thus are "clean". In the following two sections we separately investigate the impacts of the two conditions.

3 On the Importance of Identity Skip Connections

Let's consider a simple modification, $h(\mathbf{x}_l) = \lambda_l \mathbf{x}_l$, to break the identity shortcut:

$$\mathbf{x}_{l+1} = \lambda_l \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \tag{6}$$

where λ_l is a modulating scalar (for simplicity we still assume f is identity). Recursively applying this formulation we obtain an equation similar to Eqn. (4): $\mathbf{x}_L = (\prod_{i=l}^{L-1}\lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1}(\prod_{j=i+1}^{L-1}\lambda_j)\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$, or simply:

$$\mathbf{x}_L = (\prod_{i=l}^{L-1}\lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1}\hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i), \tag{7}$$

where the notation $\hat{\mathcal{F}}$ absorbs the scalars into the residual functions. Similar to Eqn.(5), we have backpropagation of the following form:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left((\prod_{i=l}^{L-1}\lambda_i) + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i)\right). \tag{8}$$

Unlike Eqn.(5), in Eqn.(8) the first additive term is modulated by a factor $\prod_{i=l}^{L-1}\lambda_i$. For an extremely deep network (L is large), if $\lambda_i > 1$ for all i, this factor can be exponentially large; if $\lambda_i < 1$ for all i, this factor can be exponentially small and vanish, which blocks the backpropagated signal from the shortcut and forces it to flow through the weight layers. This results in optimization difficulties as we show by experiments.

In the above analysis, the original identity skip connection in Eqn.(3) is replaced with a simple scaling $h(\mathbf{x}_l) = \lambda_l \mathbf{x}_l$. If the skip connection $h(\mathbf{x}_l)$ represents more complicated transforms (such as gating and 1×1 convolutions), in Eqn.(8) the first term becomes $\prod_{i=l}^{L-1}h'_i$ where h' is the derivative of h. This product may also impede information propagation and hamper the training procedure as witnessed in the following experiments.

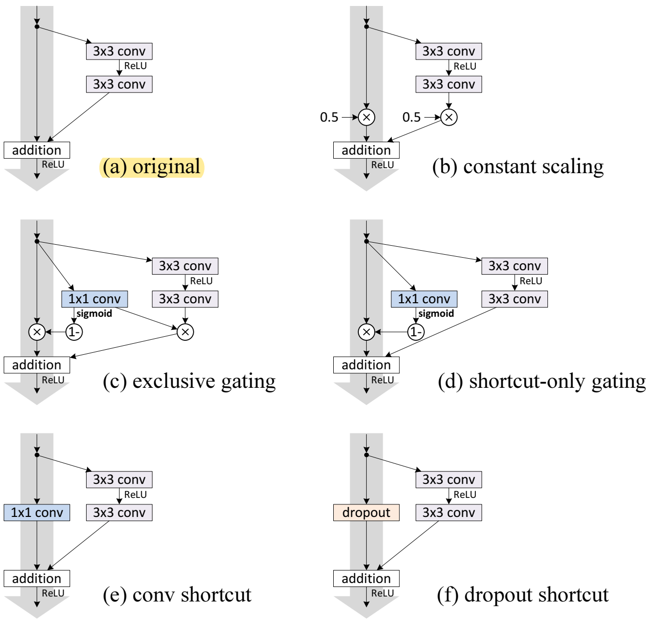

Figure 2. Various types of shortcut connections used in Table 1. The grey arrows indicate the easiest paths for the information to propagate. The shortcut connections in (b-f) are impeded by different components. For simplifying illustrations we do not display the BN layers, which are adopted right after the weight layers for all units here.

3.1 Experiments on Skip Connections

We experiment with the 110-layer ResNet as presented in [1] on CIFAR-10 [10]. This extremely deep ResNet-110 has 54 two-layer Residual Units (consisting of 3×3 convolutional layers) and is challenging for optimization. Our implementation details (see appendix) are the same as [1]. Throughout this paper we report the median accuracy of **5 runs** for each architecture on CIFAR, reducing the impacts of random variations.

Though our above analysis is driven by identity f, the experiments in this section are all based on $f = \text{ReLU}$ as in [1]; we address identity f in the next section. Our baseline ResNet-110 has 6.61% error on the test set. The comparisons of other variants (Fig. 2 and Table 1) are summarized as follows:

Constant scaling. We set $\lambda = 0.5$ for all shortcuts (Fig. 2(b)). We further study two cases of scaling $\mathcal{F}$: (i) $\mathcal{F}$ is not scaled; or (ii) $\mathcal{F}$ is scaled by a constant scalar of $1-\lambda = 0.5$, which is similar to the highway gating [6,7] but with frozen gates. The former case does not converge well; the latter is able to converge, but the test error (Table 1, 12.35%) is substantially higher than the original ResNet-110. Fig 3(a) shows that the training error is higher than that of the original ResNet-110, suggesting that the optimization has difficulties when the shortcut signal is scaled down.

Table 1. Classification error on the CIFAR-10 test set using ResNet-110 [1], with different types of shortcut connections applied to all Residual Units. We report "fail" when the test error is higher than 20%.

case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1-g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1-g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1-g(\mathbf{x})$	1	12.86	init b_g=0
		$1-g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

Exclusive gating. Following the Highway Networks [6,7] that adopt a gating mechanism [5], we consider a gating function $g(\mathbf{x}) = \sigma(\mathrm{W}_g\mathbf{x} + b_g)$ where a transform is represented by weights W_g and biases b_g followed by the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. In a convolutional network $g(\mathbf{x})$ is realized by a 1×1 convolutional layer. The gating function modulates the signal by element-wise multiplication.

We investigate the "exclusive" gates as used in [6,7] — the $\mathcal{F}$ path is scaled by $g(\mathbf{x})$ and the shortcut path is scaled by $1-g(\mathbf{x})$. See Fig 2(c). We find that the initialization of the biases b_g is critical for training gated models, and following the guidelines[2] in [6,7], we conduct hyper-parameter search on the initial value of b_g in the range of 0 to -10 with a decrement step of -1 on the training set by cross-validation. The best value (-6 here) is then used for training on the training set, leading to a test result of 8.70% (Table 1), which still lags far behind the ResNet-110 baseline. Fig 3(b) shows the training curves. Table 1 also reports the results of using other initialized values, noting that the exclusive gating network does not converge to a good solution when b_g is not appropriately initialized.

The impact of the exclusive gating mechanism is two-fold. When $1 - g(\mathbf{x})$ approaches 1, the gated shortcut connections are closer to identity which helps information propagation; but in this case $g(\mathbf{x})$ approaches 0 and suppresses the function $\mathcal{F}$. To isolate the effects of the gating functions on the shortcut path alone, we investigate a non-exclusive gating mechanism in the next.

Shortcut-only gating. In this case the function $\mathcal{F}$ is not scaled; only the shortcut path is gated by $1-g(\mathbf{x})$. See Fig 2(d). The initialized value of b_g is still essential in this case. When the initialized b_g is 0 (so initially the expectation of $1 - g(\mathbf{x})$ is 0.5), the network converges to a poor result of 12.86% (Table 1). This is also caused by higher training error (Fig 3(c)).

[2] See also: `people.idsia.ch/~rupesh/very_deep_learning/` by [6,7].

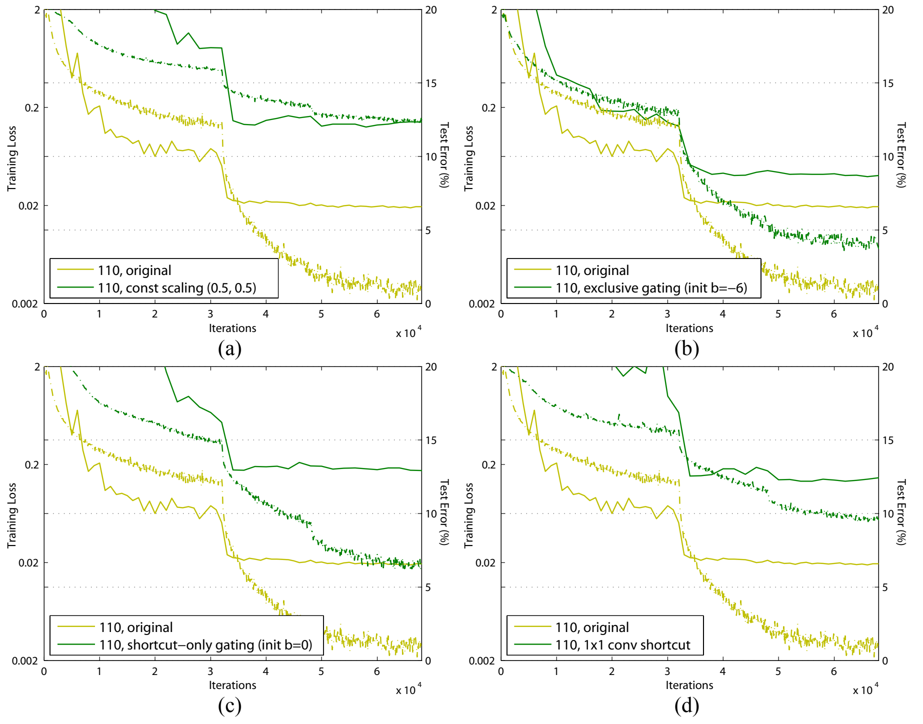

Figure 3. Training curves on CIFAR-10 of various shortcuts. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left).

When the initialized b_g is very negatively biased (*e.g.*, -6), the value of $1-g(\mathbf{x})$ is closer to 1 and the shortcut connection is nearly an identity mapping. Therefore, the result (6.91%, Table 1) is much closer to the ResNet-110 baseline.

1×1 convolutional shortcut. Next we experiment with 1×1 convolutional shortcut connections that replace the identity. This option has been investigated in [1] (known as option C) on a 34-layer ResNet (16 Residual Units) and shows good results, suggesting that 1×1 shortcut connections could be useful. But we find that this is not the case when there are many Residual Units. The 110-layer ResNet has a poorer result (12.22%, Table 1) when using 1×1 convolutional shortcuts. Again, the training error becomes higher (Fig 3(d)). When stacking so many Residual Units (54 for ResNet-110), even the shortest path may still impede signal propagation. We witnessed similar phenomena on ImageNet with ResNet-101 when using 1×1 convolutional shortcuts.

Dropout shortcut. Last we experiment with dropout [11] (at a ratio of 0.5) which we adopt on the output of the identity shortcut (Fig. 2(f)). The network fails to converge to a good solution. Dropout statistically imposes a scale of λ with an expectation of 0.5 on the shortcut, and similar to constant scaling by 0.5, it impedes signal propagation.

Table 2. Classification error (%) on the CIFAR-10 test set using different activation functions.

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

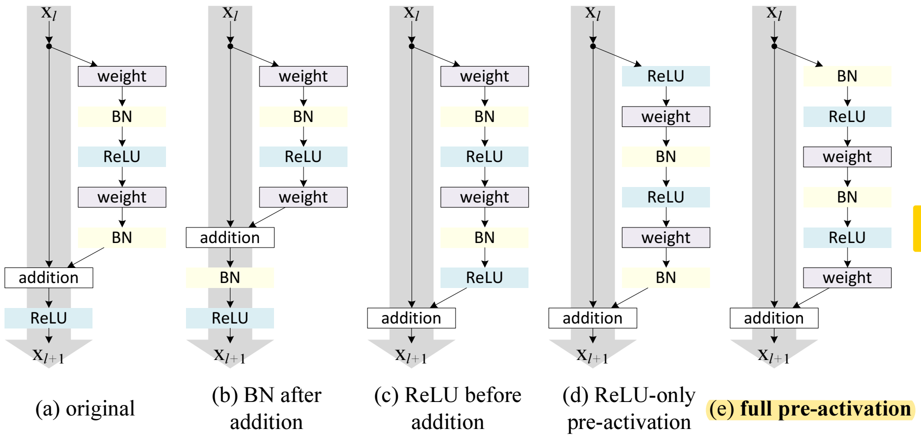

Figure 4. Various usages of activation in Table 2. All these units consist of the same components — only the orders are different.

3.2 Discussions

As indicated by the grey arrows in Fig. 2, the shortcut connections are the most direct paths for the information to propagate. *Multiplicative* manipulations (scaling, gating, 1×1 convolutions, and dropout) on the shortcuts can hamper information propagation and lead to optimization problems.

It is noteworthy that the gating and 1×1 convolutional shortcuts introduce more parameters, and should have stronger *representational* abilities than identity shortcuts. In fact, the shortcut-only gating and 1×1 convolution cover the solution space of identity shortcuts (*i.e.*, they could be optimized as identity shortcuts). However, their training error is higher than that of identity shortcuts, indicating that the degradation of these models is caused by optimization issues, instead of representational abilities.

4 On the Usage of Activation Functions

Experiments in the above section support the analysis in Eqn.(5) and Eqn.(8), both being derived under the assumption that the after-addition activation f

is the identity mapping. But in the above experiments f is ReLU as designed in [1], so Eqn.(5) and (8) are approximate in the above experiments. Next we investigate the impact of f.

We want to make f an identity mapping, which is done by re-arranging the activation functions (ReLU and/or BN). The original Residual Unit in [1] has a shape in Fig. 4(a) — BN is used after each weight layer, and ReLU is adopted after BN except that the last ReLU in a Residual Unit is after element-wise addition (f = ReLU). Fig. 4(b-e) show the alternatives we investigated, explained as following.

4.1 Experiments on Activation

In this section we experiment with ResNet-110 and a 164-layer *Bottleneck* [1] architecture (denoted as ResNet-164). A bottleneck Residual Unit consist of a 1×1 layer for reducing dimension, a 3×3 layer, and a 1×1 layer for restoring dimension. As designed in [1], its computational complexity is similar to the two-3×3 Residual Unit. More details are in the appendix. The baseline ResNet-164 has a competitive result of 5.93% on CIFAR-10 (Table 2).

BN after addition. Before turning f into an identity mapping, we go the opposite way by adopting BN after addition (Fig. 4(b)). In this case f involves BN and ReLU. The results become considerably worse than the baseline (Table 2). Unlike the original design, now the BN layer alters the signal that passes through the shortcut and impedes information propagation, as reflected by the difficulties on reducing training loss at the beginning of training (Fib. 6 left).

ReLU before addition. A naïve choice of making f into an identity mapping is to move the ReLU before addition (Fig. 4(c)). However, this leads to a *non-negative* output from the transform $\mathcal{F}$, while intuitively a "residual" function should take values in $(-\infty, +\infty)$. As a result, the forward propagated signal is monotonically increasing. This may impact the representational ability, and the result is worse (7.84%, Table 2) than the baseline. We expect to have a residual function taking values in $(-\infty, +\infty)$. This condition is satisfied by other Residual Units including the following ones.

Post-activation or pre-activation? In the original design (Eqn.(1) and Eqn.(2)), the activation $\mathbf{x}_{l+1} = f(\mathbf{y}_l)$ affects *both paths* in the *next* Residual Unit: $\mathbf{y}_{l+1} = f(\mathbf{y}_l) + \mathcal{F}(f(\mathbf{y}_l), \mathcal{W}_{l+1})$. Next we develop an *asymmetric* form where an activation $\hat{f}$ only affects the $\mathcal{F}$ path: $\mathbf{y}_{l+1} = \mathbf{y}_l + \mathcal{F}(\hat{f}(\mathbf{y}_l), \mathcal{W}_{l+1})$, for any l (Fig. 5 (a) to (b)). By renaming the notations, we have the following form:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\hat{f}(\mathbf{x}_l), \mathcal{W}_l), . \tag{9}$$

It is easy to see that Eqn.(9) is similar to Eqn.(4), and can enable a backward formulation similar to Eqn.(5). For this new Residual Unit as in Eqn.(9), the new after-addition activation becomes an identity mapping. This design means that if a new after-addition activation $\hat{f}$ is asymmetrically adopted, it is equivalent to recasting $\hat{f}$ as the *pre-activation* of the next Residual Unit. This is illustrated in Fig. 5.

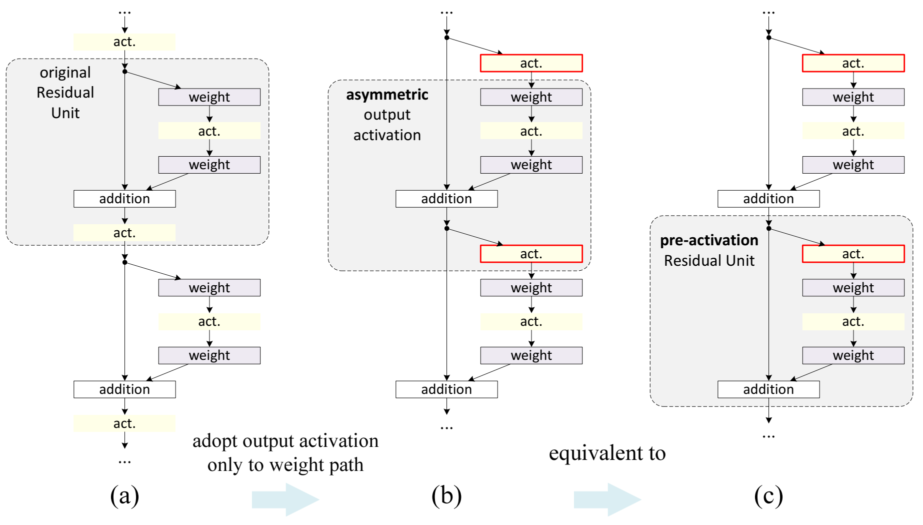
original Residual Unit
asymmetric output activation
pre-activation Residual Unit
act.
weight
addition
adopt output activation only to weight path
equivalent to
(a)
(b)
(c)

Figure 5. Using asymmetric after-addition activation is equivalent to constructing a *pre-activation* Residual Unit.

Table 3. Classification error (%) on the CIFAR-10/100 test set using the original Residual Units and our pre-activation Residual Units.

dataset	network	baseline unit	pre-activation unit
CIFAR-10	ResNet-110 (1layer skip)	9.90	8.91
	ResNet-110	6.61	6.37
	ResNet-164	5.93	5.46
	ResNet-1001	7.61	4.92
CIFAR-100	ResNet-164	25.16	24.33
	ResNet-1001	27.82	22.71

The distinction between post-activation/pre-activation is caused by the presence of the element-wise *addition*. For a plain network that has N layers, there are $N-1$ activations (BN/ReLU), and it does not matter whether we think of them as post- or pre-activations. But for branched layers merged by addition, the position of activation matters.

We experiment with two such designs: (i) ReLU-only pre-activation (Fig. 4(d)), and (ii) full pre-activation (Fig. 4(e)) where BN and ReLU are both adopted before weight layers. Table 2 shows that the ReLU-only pre-activation performs very similar to the baseline on ResNet-110/164. This ReLU layer is not used in conjunction with a BN layer, and may not enjoy the benefits of BN [8].

Somehow surprisingly, when BN and ReLU are both used as pre-activation, the results are improved by healthy margins (Table 2 and Table 3). In Table 3 we report results using various architectures: (i) ResNet-110, (ii) ResNet-164, (iii) a 110-layer ResNet architecture in which each shortcut skips only 1 layer (*i.e.*,

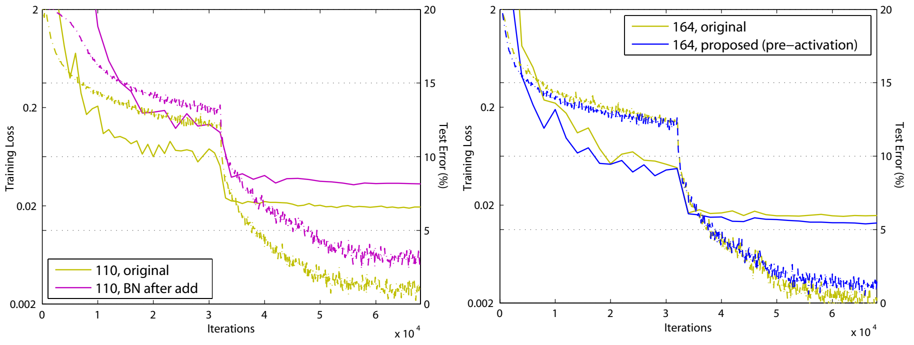

Figure 6. Training curves on CIFAR-10. **Left**: BN after addition (Fig. 4(b)) using ResNet-110. **Right**: pre-activation unit (Fig. 4(e)) on ResNet-164. Solid lines denote test error, and dashed lines denote training loss.

a Residual Unit has only 1 layer), denoted as "ResNet-110(1layer)", and (iv) a 1001-layer bottleneck architecture that has 333 Residual Units (111 on each feature map size), denoted as "ResNet-1001". We also experiment on CIFAR-100. Table 3 shows that our "pre-activation" models are consistently better than the baseline counterparts. We analyze these results in the following.

4.2 Analysis

We find the impact of pre-activation is twofold. First, the optimization is further eased (comparing with the baseline ResNet) because f is an identity mapping. Second, using BN as pre-activation improves regularization of the models.

Ease of optimization. This effect is particularly obvious when training the *1001-layer* ResNet. Fig. 1 shows the curves. Using the original design in [1], the training error is reduced very slowly at the beginning of training. For f = ReLU, the signal is impacted if it is negative, and when there are many Residual Units, this effect becomes prominent and Eqn.(3) (so Eqn.(5)) is not a good approximation. On the other hand, when f is an identity mapping, the signal can be propagated directly between any two units. Our 1001-layer network reduces the training loss very quickly (Fig. 1). It also achieves the lowest loss among all models we investigated, suggesting the success of optimization.

We also find that the impact of f = ReLU is not severe when the ResNet has fewer layers (*e.g.*, 164 in Fig. 6(right)). The training curve seems to suffer a little bit at the beginning of training, but goes into a healthy status soon. By monitoring the responses we observe that this is because after some training, the weights are adjusted into a status such that $\mathbf{y}_l$ in Eqn.(1) is more frequently above zero and f does not truncate it ($\mathbf{x}_l$ is always non-negative due to the previous ReLU, so $\mathbf{y}_l$ is below zero only when the magnitude of $\mathcal{F}$ is very negative). The truncation, however, is more frequent when there are 1000 layers.

Table 4. Comparisons with state-of-the-art methods on CIFAR-10 and CIFAR-100 using "*moderate data augmentation*" (flip/translation), except for ELU [12] with no augmentation. Better results of [13,14] have been reported using stronger data augmentation and ensembling. For the ResNets we also report the number of parameters. Our results are the median of 5 runs with mean±std in the brackets. All ResNets results are obtained with a mini-batch size of 128 except † with a mini-batch size of 64 (code available at https://github.com/KaimingHe/resnet-1k-layers).

CIFAR-10	error (%)
NIN [15]	8.81
DSN [16]	8.22
FitNet [17]	8.39
Highway [7]	7.72
All-CNN [14]	7.25
ELU [12]	6.55
FitResNet, LSUV [18]	5.84
ResNet-110 [1] (1.7M)	6.61
ResNet-1202 [1] (19.4M)	7.93
ResNet-164 [ours] (1.7M)	5.46
ResNet-1001 [ours] (10.2M)	4.92 (4.89±0.14)
ResNet-1001 [ours] (10.2M)†	**4.62** (4.69±0.20)

CIFAR-100	error (%)
NIN [15]	35.68
DSN [16]	34.57
FitNet [17]	35.04
Highway [7]	32.39
All-CNN [14]	33.71
ELU [12]	24.28
FitNet, LSUV [18]	27.66
ResNet-164 [1] (1.7M)	25.16
ResNet-1001 [1] (10.2M)	27.82
ResNet-164 [ours] (1.7M)	24.33
ResNet-1001 [ours] (10.2M)	**22.71** (22.68±0.22)

Reducing overfitting. Another impact of using the proposed pre-activation unit is on regularization, as shown in Fig. 6 (right). The pre-activation version reaches slightly higher training loss at convergence, but produces lower test error. This phenomenon is observed on ResNet-110, ResNet-110(1-layer), and ResNet-164 on both CIFAR-10 and 100. This is presumably caused by BN's regularization effect [8]. In the original Residual Unit (Fig. 4(a)), although the BN normalizes the signal, this is soon added to the shortcut and thus the merged signal is not normalized. This unnormalized signal is then used as the input of the next weight layer. On the contrary, in our pre-activation version, the inputs to all weight layers have been normalized.

5 Results

Comparisons on CIFAR-10/100. Table 4 compares the state-of-the-art methods on CIFAR-10/100, where we achieve competitive results. We note that we do not specially tailor the network width or filter sizes, nor use regularization techniques (such as dropout) which are very effective for these small datasets. We obtain these results via a simple but essential concept — going deeper. These results demonstrate the potential of *pushing the limits of depth.*

Comparisons on ImageNet. Next we report experimental results on the 1000-class ImageNet dataset [3]. We have done preliminary experiments using the skip connections studied in Fig. 2 & 3 on ImageNet with ResNet-101 [1], and observed similar optimization difficulties. The training error of these non-identity shortcut networks is obviously higher than the original ResNet at the first learning rate

Table 5. Comparisons of single-crop error on the ILSVRC 2012 validation set. All ResNets are trained using the same hyper-parameters and implementations as [1]). Our Residual Units are the full pre-activation version (Fig. 4(e)). †: code/model available at https://github.com/facebook/fb.resnet.torch/tree/master/pretrained, using scale and aspect ratio augmentation in [20].

method	augmentation	train crop	test crop	top-1	top-5
ResNet-152, original Residual Unit [1]	scale	224×224	224×224	23.0	6.7
ResNet-152, original Residual Unit [1]	scale	224×224	320×320	21.3	5.5
ResNet-152, **pre-act** Residual Unit	scale	224×224	320×320	21.1	5.5
ResNet-200, original Residual Unit [1]	scale	224×224	320×320	21.8	6.0
ResNet-200, **pre-act** Residual Unit	scale	224×224	320×320	**20.7**	**5.3**
ResNet-200, **pre-act** Residual Unit	scale+asp ratio	224×224	320×320	**20.1**†	**4.8**†
Inception v3 [19]	scale+asp ratio	299×299	299×299	21.2	5.6

(similar to Fig. 3), and we decided to halt training due to limited resources. But we did finish a "BN after addition" version (Fig. 4(b)) of ResNet-101 on ImageNet and observed higher training loss and validation error. This model's single-crop (224×224) validation error is 24.6%/7.5%, *vs.* the original ResNet-101's 23.6%/7.1%. This is in line with the results on CIFAR in Fig. 6 (left).

Table 5 shows the results of ResNet-152 [1] and ResNet-200[3], all trained from scratch. We notice that the original ResNet paper [1] trained the models using scale jittering with shorter side $s \in [256, 480]$, and so the test of a 224×224 crop on $s = 256$ (as did in [1]) is negatively biased. Instead, we test a single 320×320 crop from $s = 320$, for all original and our ResNets. Even though the ResNets are trained on smaller crops, they can be easily tested on larger crops because the ResNets are fully convolutional by design. This size is also close to 299×299 used by Inception v3 [19], allowing a fairer comparison.

The original ResNet-152 [1] has top-1 error of 21.3% on a 320×320 crop, and our pre-activation counterpart has 21.1%. The gain is not big on ResNet-152 because this model has not shown severe generalization difficulties. However, the original ResNet-200 has an error rate of 21.8%, higher than the baseline ResNet-152. But we find that the original ResNet-200 has *lower* training error than ResNet-152, suggesting that it suffers from overfitting.

Our pre-activation ResNet-200 has an error rate of 20.7%, which is **1.1%** lower than the baseline ResNet-200 and also lower than the two versions of ResNet-152. When using the scale and aspect ratio augmentation of [20,19], our ResNet-200 has a result better than Inception v3 [19] (Table 5). Concurrent with our work, an Inception-ResNet-v2 model [21] achieves a single-crop result of 19.9%/4.9%. We expect our observations and the proposed Residual Unit will help this type and generally other types of ResNets.

Computational Cost. Our models' computational complexity is linear on

[3] The ResNet-200 has 16 more 3-layer bottleneck Residual Units than ResNet-152, which are added on the feature map of 28×28.

depth (so a 1001-layer net is ∼10× complex of a 100-layer net). On CIFAR, ResNet-1001 takes about 27 hours to train on 2 GPUs; on ImageNet, ResNet-200 takes about 3 weeks to train on 8 GPUs (on par with VGG nets [22]).

6 Conclusions

This paper investigates the propagation formulations behind the connection mechanisms of deep residual networks. Our derivations imply that identity shortcut connections and identity after-addition activation are essential for making information propagation smooth. Ablation experiments demonstrate phenomena that are consistent with our derivations. We also present 1000-layer deep networks that can be easily trained and achieve improved accuracy.

Appendix: Implementation Details The implementation details and hyper-parameters are the same as those in [1]. On CIFAR we use only the translation and flipping augmentation in [1] for training. The learning rate starts from 0.1, and is divided by 10 at 32k and 48k iterations. Following [1], for all CIFAR experiments we warm up the training by using a smaller learning rate of 0.01 at the beginning 400 iterations and go back to 0.1 after that, although we remark that this is not necessary for our proposed Residual Unit. The mini-batch size is 128 on 2 GPUs (64 each), the weight decay is 0.0001, the momentum is 0.9, and the weights are initialized as in [23].

On ImageNet, we train the models using the same data augmentation as in [1]. The learning rate starts from 0.1 (no warming up), and is divided by 10 at 30 and 60 epochs. The mini-batch size is 256 on 8 GPUs (32 each). The weight decay, momentum, and weight initialization are the same as above.

When using the pre-activation Residual Units (Fig. 4(d)(e) and Fig. 5), we pay special attention to the first and the last Residual Units of the entire network. For the first Residual Unit (that follows a stand-alone convolutional layer, conv_1), we adopt the first activation right after conv_1 and before splitting into two paths; for the last Residual Unit (followed by average pooling and a fully-connected classifier), we adopt an extra activation right after its element-wise addition. These two special cases are the natural outcome when we obtain the pre-activation network via the modification procedure as shown in Fig. 5.

The bottleneck Residual Units (for ResNet-164/1001 on CIFAR) are constructed following [1]. For example, a $\begin{bmatrix} 3{\times}3,\ 16 \\ 3{\times}3,\ 16 \end{bmatrix}$ unit in ResNet-110 is replaced with a $\begin{bmatrix} 1{\times}1,\ 16 \\ 3{\times}3,\ 16 \\ 1{\times}1,\ 64 \end{bmatrix}$ unit in ResNet-164, both of which have roughly the same number of parameters. For the bottleneck ResNets, when reducing the feature map size we use projection shortcuts [1] for increasing dimensions, and when pre-activation is used, these projection shortcuts are also with pre-activation.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
2. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. (2010)
3. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
4. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. (2014)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
6. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. In: ICML workshop. (2015)
7. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: NIPS. (2015)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015)
9. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation (1989)
10. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech Report (2009)
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 (2012)
12. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). In: ICLR. (2016)
13. Graham, B.: Fractional max-pooling. arXiv:1412.6071 (2014)
14. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv:1412.6806 (2014)
15. Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR. (2014)
16. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS. (2015)
17. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: ICLR. (2015)
18. Mishkin, D., Matas, J.: All you need is a good init. In: ICLR. (2016)
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR. (2016)
20. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015)
21. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv:1602.07261 (2016)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015)
23. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV. (2015)