# Deep Residual Learning for Image Recognition

박은우

# Abstract

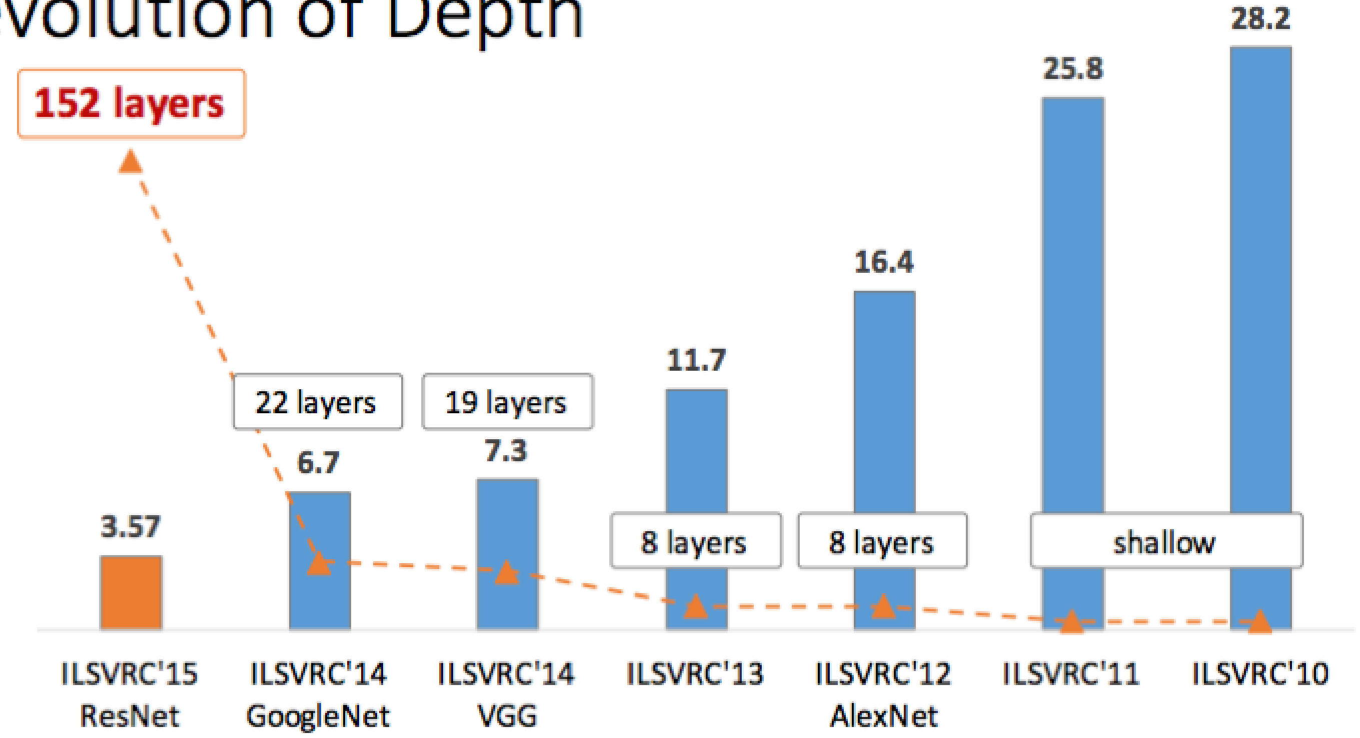
- Why “ResNet”?

To ease the training of networks that are deeper than those used previously.

- Advantage

Easier to optimize, can gain accuracy from considerably increased depth.

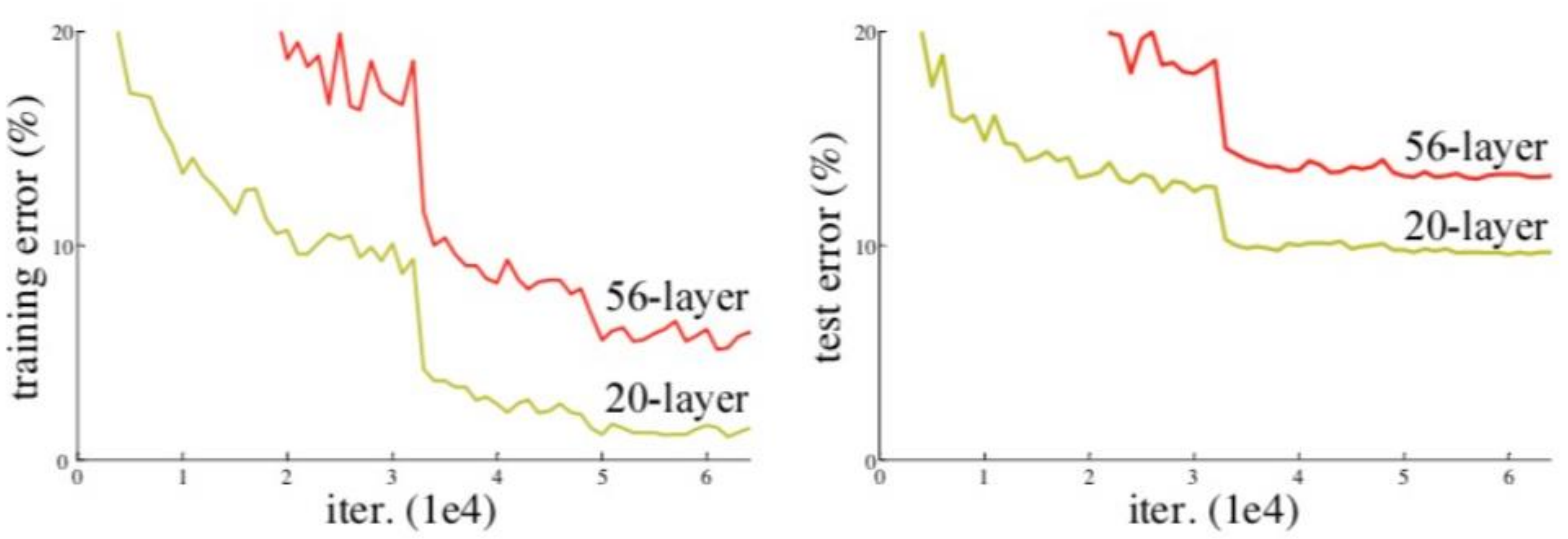
# Revolution of Depth




ImageNet Classification top-5 error (%)

# 1. Introduction

- Recent evidence reveals that network depth is of crucial importance.



- But, is learning better networks as easy as stacking more layers?

1. vanishing/exploding gradients

✓ hamper convergence from the beginning

✓ solve the problem by normalized initialization & intermediate normalization layers

2. Degradation

✓ Not by overfitting

✓ adding more layers to a suitably deep model leads to higher training error

✓ solve the problem by construction to the deeper model

: added layers are identity mapping, the other layers are copied from the learned shallower model

- Address the degradation problem by introducing a deep residual learning framework

-extra para. X

-computational complexity ↑ x

- H(x):=F(x)+x

- 출력과 입력 간 차에 대해 학습시키면 Degradation 해결 가능

- F(x)=H(x)-x를 H(x)에 근사시키는 것이 이전 모델을 optimize하는 것보다 쉽다.

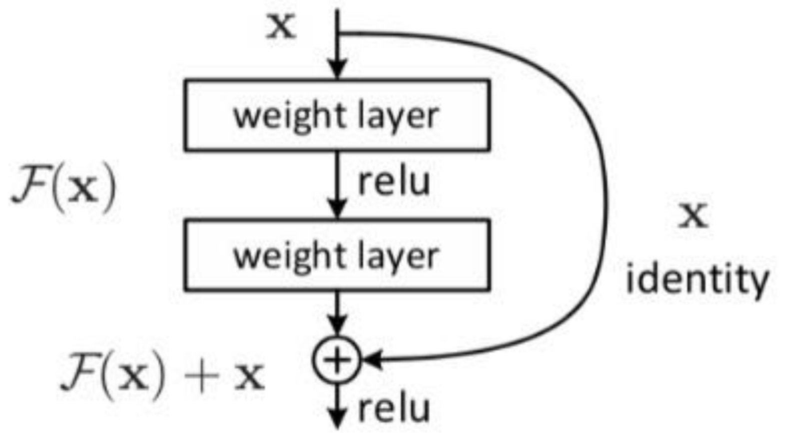


Figure 2. Residual learning: a building block.

# 2. Related work

- 이전에는 multigrid method를 사용하였음
- “highway networks” present shortcut connections with gating functions.
- our identity shortcuts are never closed, and all info is always passed through.

# 3. Deep Residual Learning

- 3.1 Residual learning

 Approximate the residual functions: H(x)-x (assuming that the input and output are of the same dimensions)

- This reformulation is motivated by the counterintuitive phenomena about the degradation problem

 상식: deeper, training error는 낮은 모델보다 더 낮을 수x

- 3.2 Identity Mapping by Shortcuts

ResNet은 layer가 적게 쌓여도 residual learning을 적용한다.

shortcut은 extra para가 필요x라는 것과 complexity 증가 x라는 것이 기존의 plain과 residual을 비교하는 데 매력적인 요소이다.

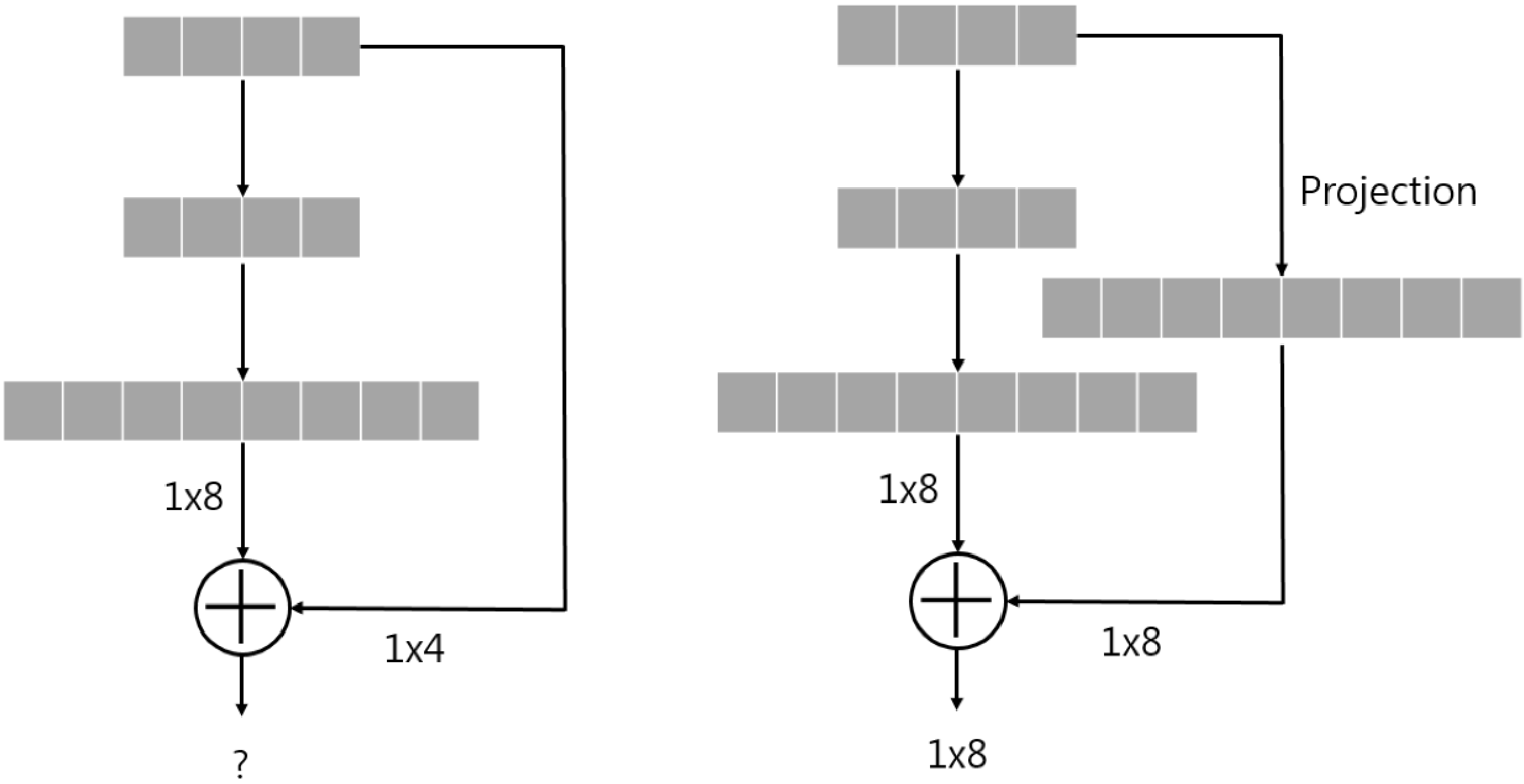
$$y = F(x, W_i) + x \qquad (1)$$

# X와 F는 차원이 같아야 함.

$$F = W_2 \sigma(W_1 x)$$

$$y = F(x, W_i) + W_s x \qquad (2)$$

# 그렇지 않을 경우 linear projection. 차원을 맞추는 데만 이용


1x8
1x4
?
Projection
1x8
1x8
1x8

- 3.3 Network Architectures

1. plain network

✓그냥 layers를 쌓음. 대부분 3x3 filters를 가짐

✓같은 크기의 output feature map가지고 같은 수 filters 가짐

✓feature map size 반으로 줄면 time-complexity를 유지하기 위해 filters의 수 2배

✓ VGG보다 filter 수 적고 complexity 낮음

2. residual network

✓ plain에 기반하여 shortcut 추가한 버전.

✓ input과 output 차원 동일하면 identity shortcut 바로 사용 가능

(1) zero-padding

(2) 차원 맞추기 위해 1x1 convolution 사용

-> stride=2

- 3.4 Implementation

[256,480] randomly sampled
batch normalization 적용
learning rate=0.1 (local min-> 1/10)
iter 60x10^4
weight decay=0.0001, momentum=0.9, dropout x
10-crop 사용

(a) Single Center Crop (b) Part of a 10-Crop

# 4. Experiments

- 4.1 ImageNet Classification

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

- 4.1.1 Plain Networks

18,34 layer, plain network

| | plain | ResNet |
|---|---|---|
| 18 layers | 27.94 | 27.88 |
| 34 layers | 28.54 | **25.03** |

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

- 4.1.2 Residual networks

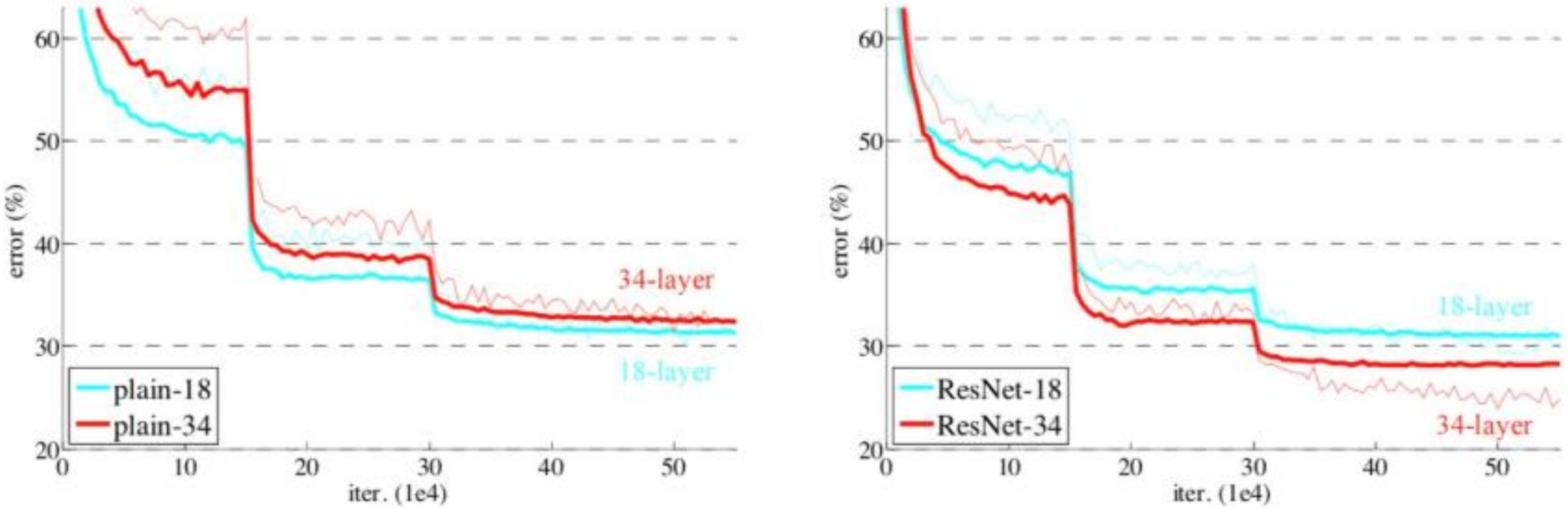



Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

- 4.1.3 Identity vs projection shortcuts

| model | top-1 err. | top-5 err. |
|---|---|---|
| VGG-16 [41] | 28.07 | 9.33 |
| GoogLeNet [44] | - | 9.15 |
| PReLU-net [13] | 24.27 | 7.38 |
| plain-34 | 28.54 | 10.02 |
| ResNet-34 A | 25.03 | 7.76 |
| ResNet-34 B | 24.52 | 7.46 |
| ResNet-34 C | 24.19 | 7.40 |
| ResNet-50 | 22.85 | 6.71 |
| ResNet-101 | 21.75 | 6.05 |
| ResNet-152 | **21.43** | **5.71** |

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

- 4.1.4 Deeper Bottlenect Architectures

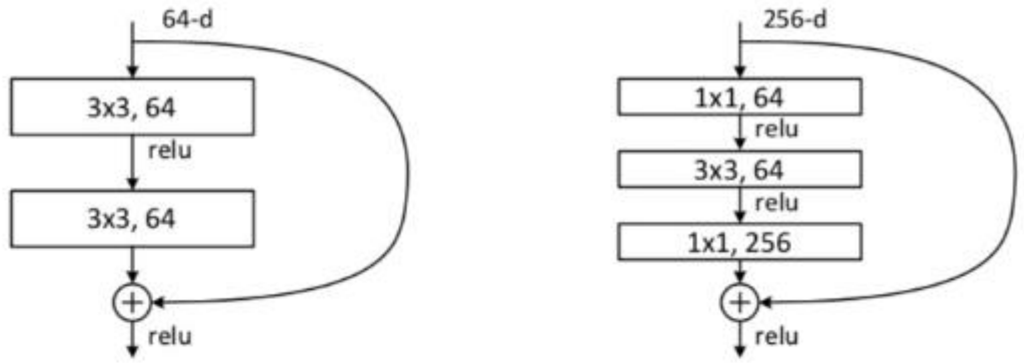



Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

- 4.1.5 50-layers ResNet

  3-layer bottlenect block을 차원 증가

- 4.1.6 101-layers and 152-layers ResNet

  depth만 증가. 좋은 성능. Degradation x

- 4.1.7 Comparisons with State-of-the-art methods

| method | top-5 err. (**test**) |
|---|---|
| VGG [41] (ILSVRC'14) | 7.32 |
| GoogLeNet [44] (ILSVRC'14) | 6.66 |
| VGG [41] (v5) | 6.8 |
| PReLU-net [13] | 4.94 |
| BN-inception [16] | 4.82 |
| **ResNet (ILSVRC'15)** | **3.57** |

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

- 4.2.1 Analysis of layer Responses

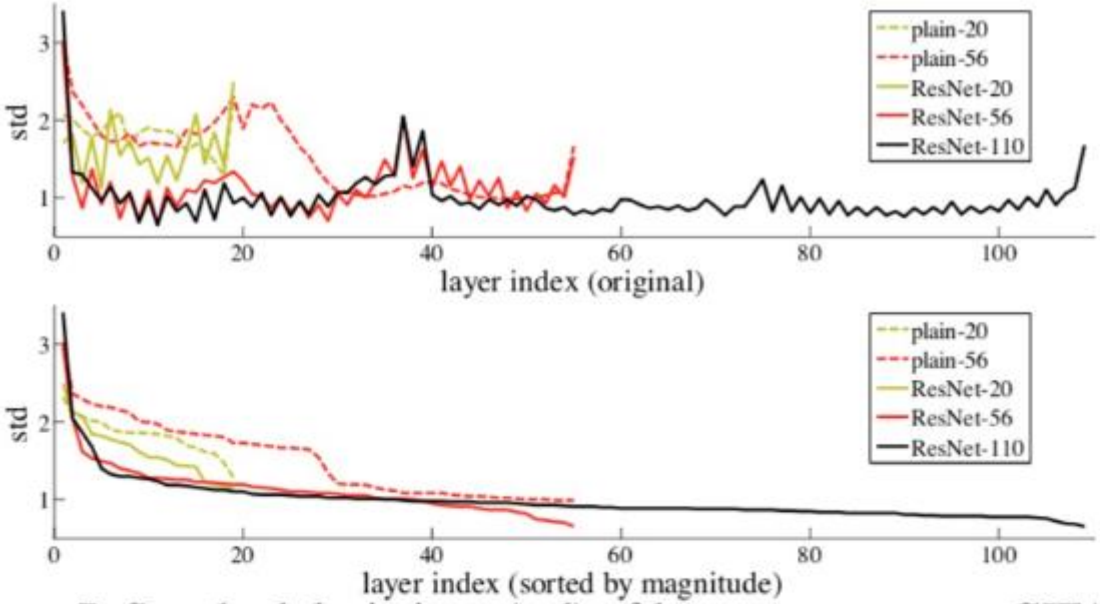
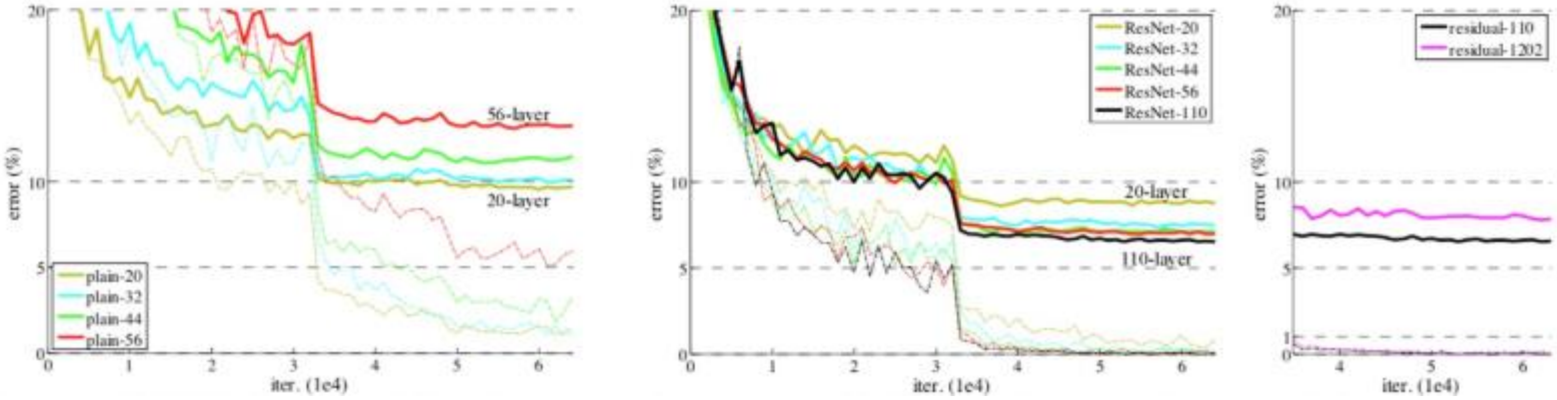



Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

• 4.2.2 Exploring over 1000 layers

| method | | | error (%) |
|---|---|---|---|
| Maxout [10] | | | 9.38 |
| NIN [25] | | | 8.81 |
| DSN [24] | | | 8.22 |
| | # layers | # params | |
| FitNet [35] | 19 | 2.5M | 8.39 |
| Highway [42, 43] | 19 | 2.3M | 7.54 (7.72±0.16) |
| Highway [42, 43] | 32 | 1.25M | 8.80 |
| ResNet | 20 | 0.27M | 8.75 |
| ResNet | 32 | 0.46M | 7.51 |
| ResNet | 44 | 0.66M | 7.17 |
| ResNet | 56 | 0.85M | 6.97 |
| ResNet | 110 | 1.7M | **6.43** (6.61±0.16) |
| ResNet | 1202 | 19.4M | 7.93 |



- 4.3 Object Detection on PASCAL and MS COCO

| training data | 07+12 | 07++12 |
|---|---|---|
| test data | VOC 07 test | VOC 12 test |
| VGG-16 | 73.2 | 70.4 |
| ResNet-101 | **76.4** | **73.8** |

Table 7. Object detection mAP (%) on the PASCAL VOC 2007/2012 test sets using **baseline** Faster R-CNN. See also Table 10 and 11 for better results.

| metric | mAP@.5 | mAP@[.5, .95] |
|---|---|---|
| VGG-16 | 41.5 | 21.2 |
| ResNet-101 | **48.4** | **27.2** |

Table 8. Object detection mAP (%) on the COCO validation set using **baseline** Faster R-CNN. See also Table 9 for better results.