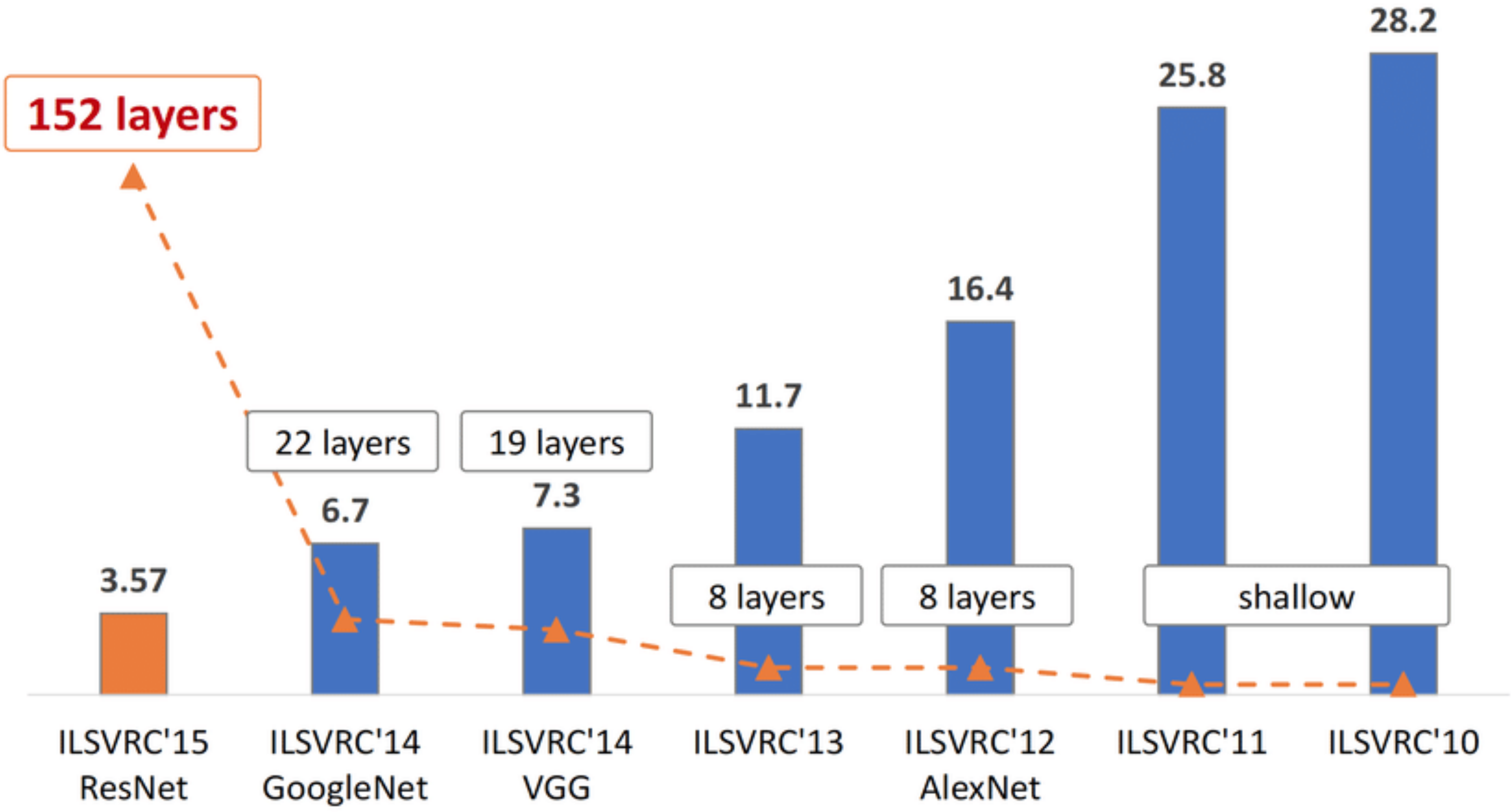
ResNet:Deep Residual Learning for Image Recognition
[He et al., CVPR 2015]

이하연

ResNet: Residual Network



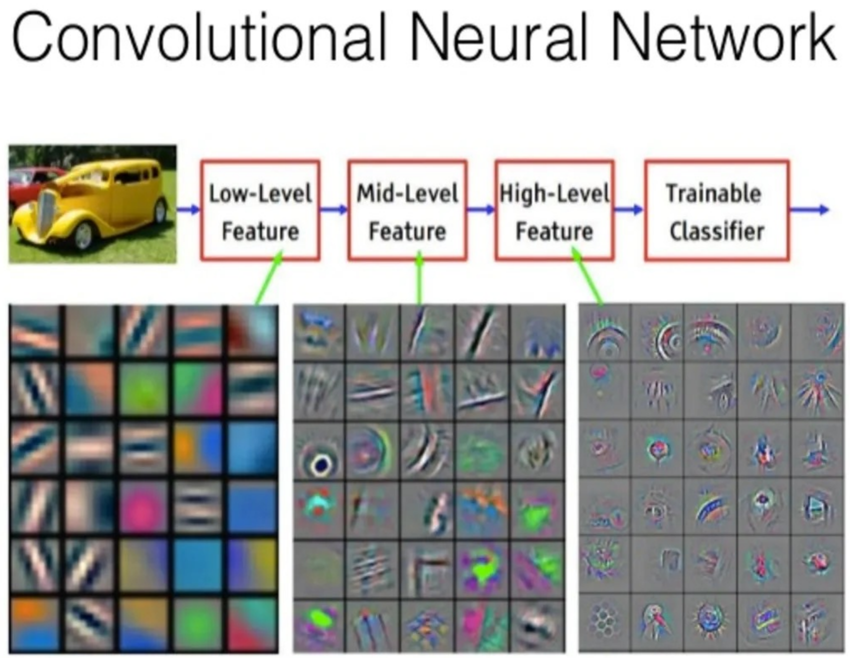
The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge from 2010 to 2015.

Deep convolutional neural network

깊은 네트워크는 low/mid/high level의 feature와 classifiers를 an end-to-end multilayer 방식으로 자연스럽게 통합.

이러한 'levels' of the features는 stacked layer의 깊이에 의해 풍부해짐.

- Larger receptive fields
- More capacity and non-linearity



Feature Visualization of Convnet trained on ImageNet from [Zeiler & Fergus 2013]

Is learning better networks as easy as stacking more layers?

1. problem of vanishing/exploding gradients
   → normalized initialization과 intermediate normalization layer로 해결

2. **Degradation Problem**: network가 깊어질수록 accuracy가 떨어지는 문제
   → Overfitting의 문제 X, 깊은 레이어를 쌓을수록 Optimize하기 어려워지기 때문에 발생

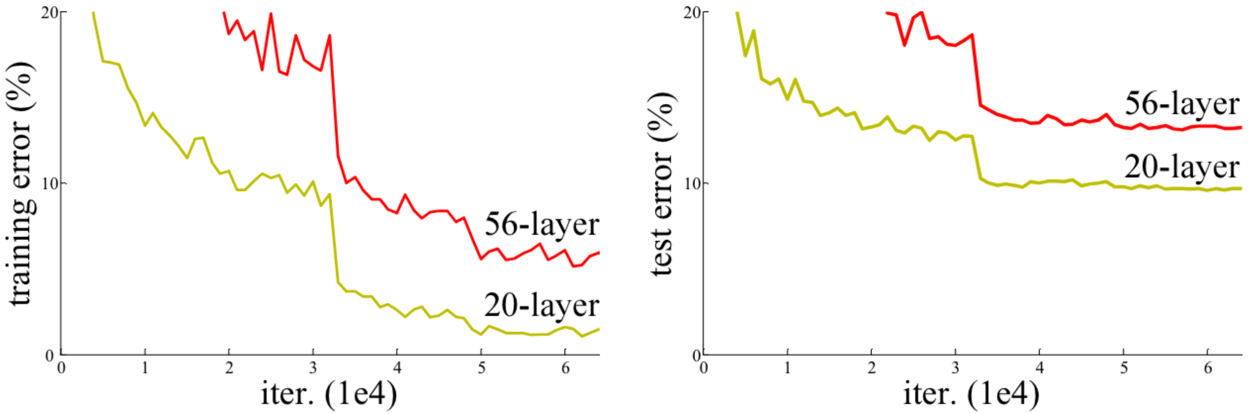


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Deep Residual Learning Framework: Shortcut Connection

기존의 neural net: underlying mapping인 H(x)를 최소화 하는 것이 목표
But, H(x)를 직접적으로 최소화 시키는 것은 어려움

A solution: residual mapping

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

→ original mapping인 H(x) = F(x) + x의 형태가 됨
→ identity shortcut connections은 추가 parameter나 computational complexity가 요구 X

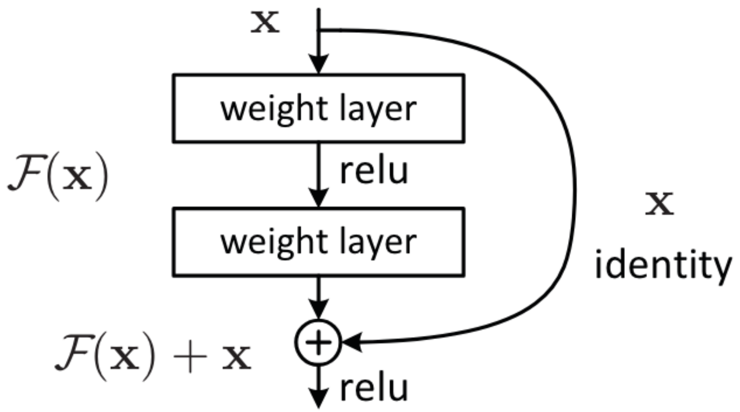


Figure 2. Residual learning: a building block.

Deep Residual Learning Framework: Shortcut Connection

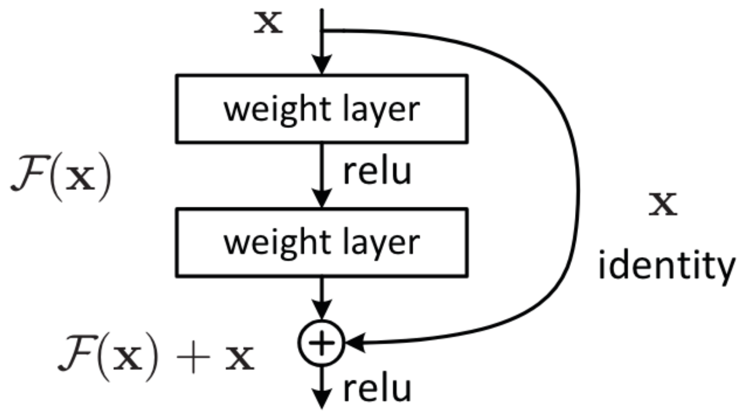


Figure 2. Residual learning: a building block.

1. H(x) = x 가 되도록 train
2. network의 output인 F(x)는 0이 되도록 train

H(x) = F(x) + x = x 이를 미분 시 미분 값이 1 이상, 따라서 모든 계층에서 gradient vanishing 현상을 해결

Identity Mapping by Shortcuts

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \quad \textbf{(1)}$$

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \quad \textbf{(2)}$$

x, y: Input/output vectors

F(x, {Wi}): residual mapping to be learned

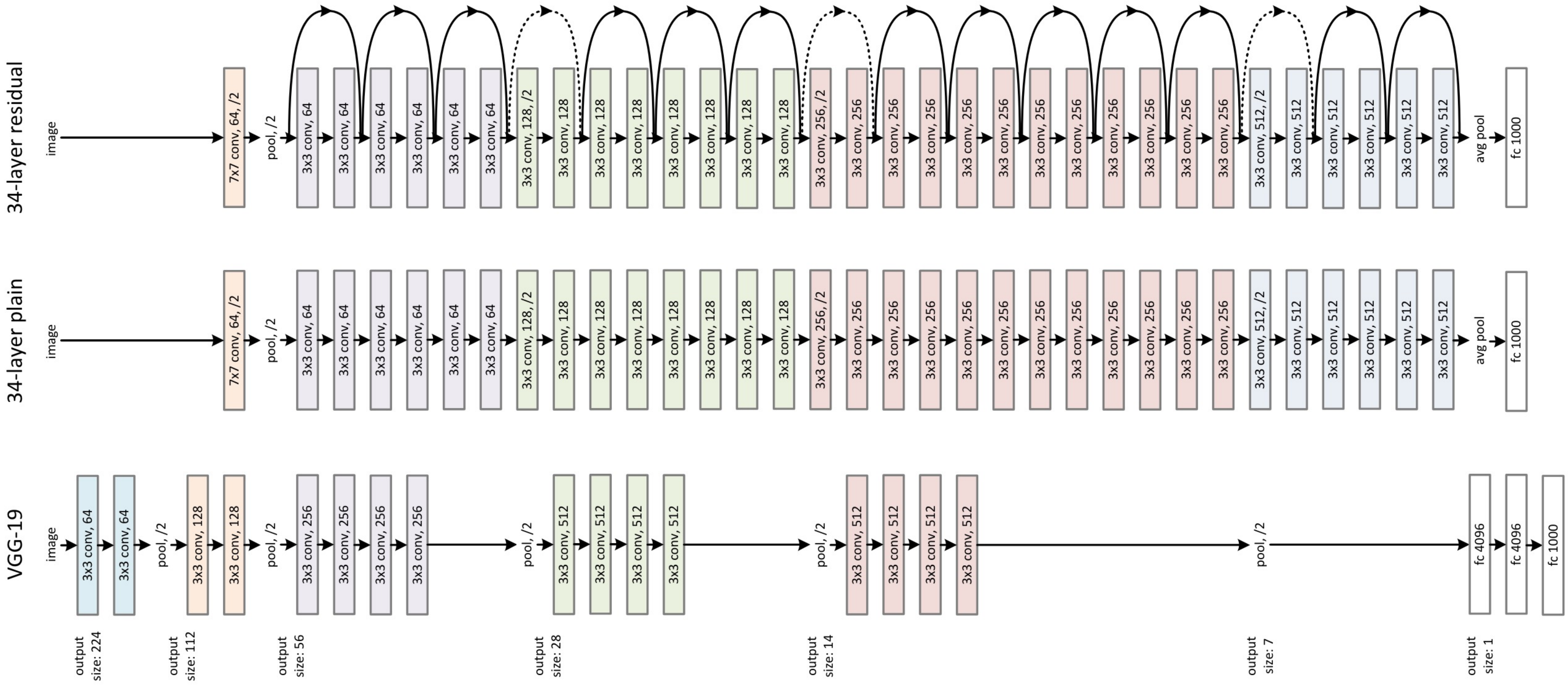
σ: ReLU

F + x: shortcut connection and element-wise addition

Ws: a square matrix used to match dimensions

F is flexible

Network Architectures



Residual Network for ImageNet

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		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Experiment: ImageNet Classification

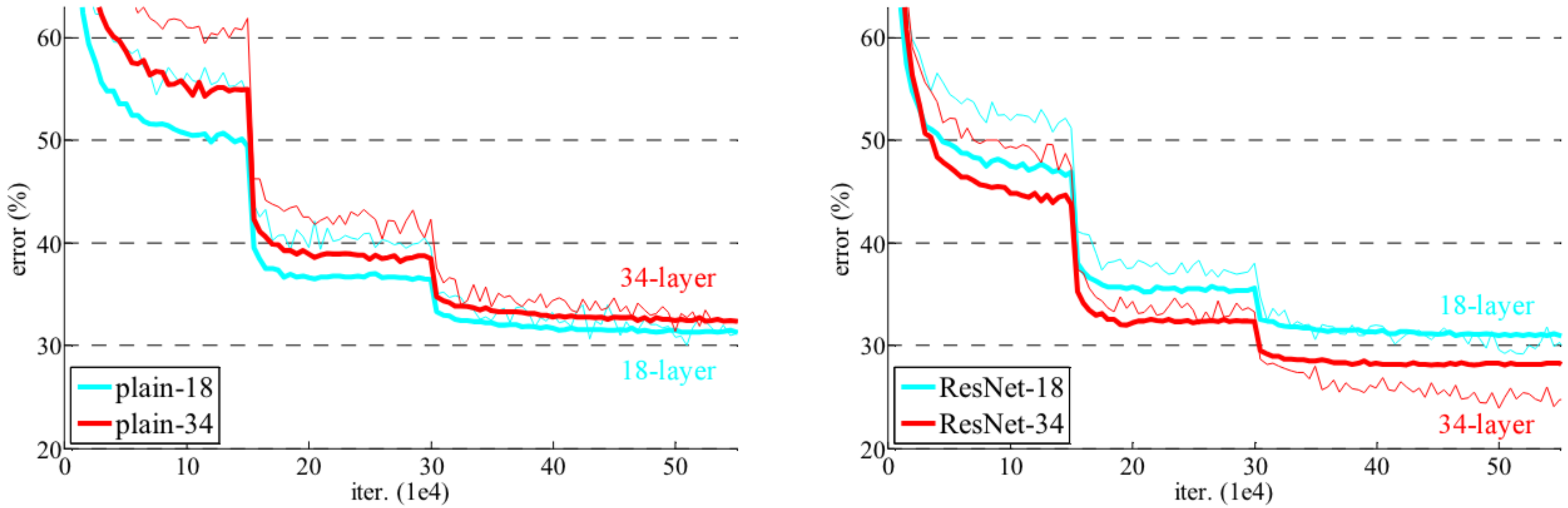


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Deeper Bottleneck Architectures

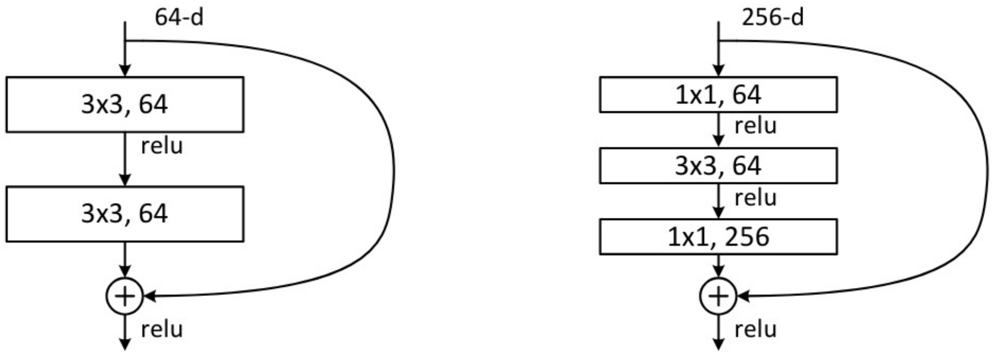


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Concerns on training time: building block → bottleneck design
residual function F: 3개의 Layer stack → 1x1, 3x3, 1x1 Conv
1x1 Conv로 input output dimension 유지

parameter free identity shortcut은 bottleneck architecture에서 중요
만약 projection으로 대체한다면 시간 복잡도, 모델의 크기가 두 배 증가
→ shortcut이 두개의 high-dimensonal 의 끝에 연결되기 때문

Analysis of Layer Responses

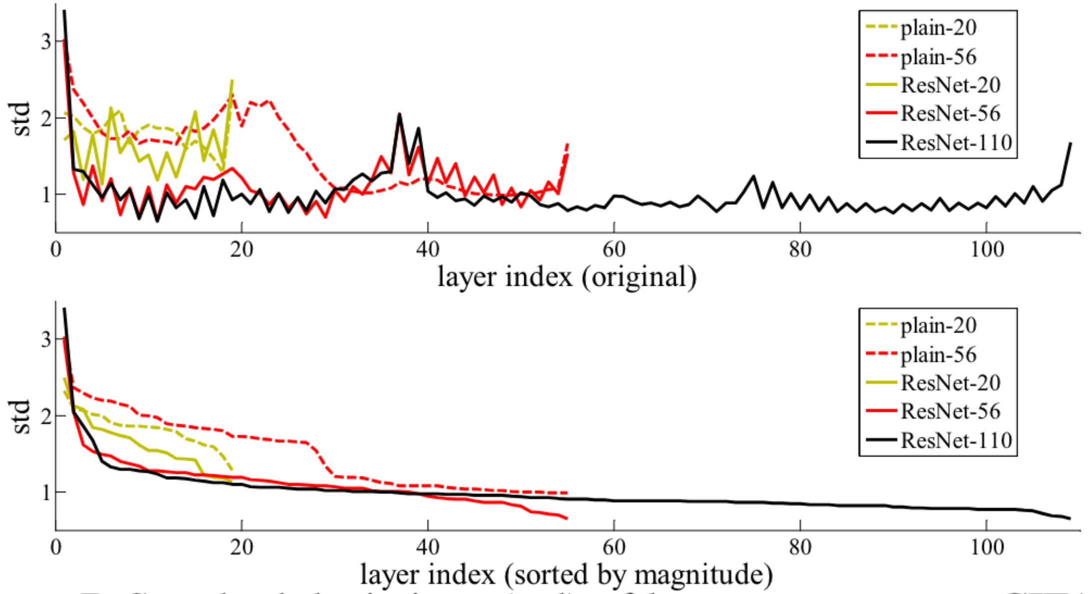


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

Responses : the outputs of each 3×3 layer, after BN and before other nonlinearity (ReLU/addition)

ResNets have generally **smaller responses** than plain counterparts.

The residual functions might be generally closer to zero than the non-residual functions.
(F(x) := H(x) - x)

References

- He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. (https://arxiv.org/pdf/1512.03385.pdf)

QnA