Convolutional Neural Network

ResNet

2022.02.21

황성아

Abstract

“ 마이크로소프트에서 개발, ILSVRC 2015 우승”

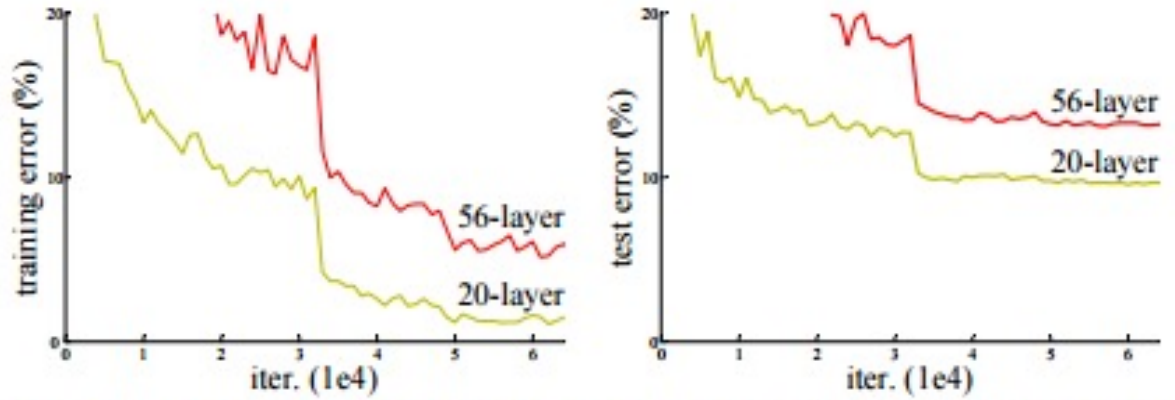


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

20층의 네트워크와 56층의 네트워크로 실험
-> 56층 레이어가 오히려 더 안좋은 성능을 보임

성능을 높이기 위해 가장 쉽게 생각하는 방법은 layer 를 깊게 쌓는 것

하지만, layer 가 너무 깊어지면 gradient vanishing(기울기 소실) 등의 문제 발생

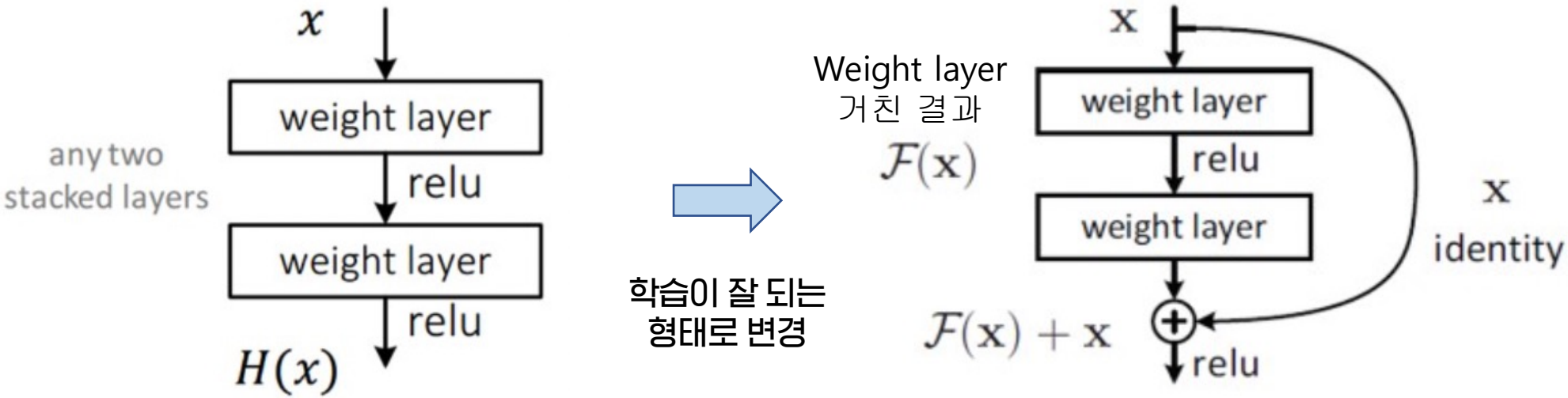
따라서, ResNet은 깊은 네트워크를 학습시키기 위해 Residual(잔차)를 학습하는 방법 적용

ResNet

Deep Residual Learning for image Recognition(CVPR 2016) [Paper]

ResNet Architecture

"Residual Learning을 이용해 네트워크의 최적화 난이도를 낮춤"



<일반 CNN>

<ResNet>

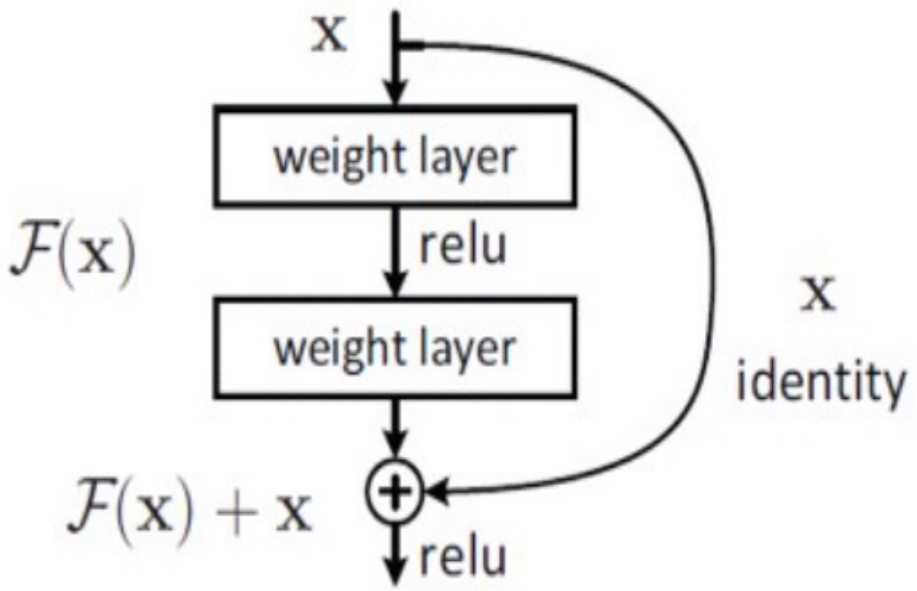
input x를 받아 2개의 weighted layer 를 거쳐 output H(x)를 내며, 다음 layer의 입력으로 적용

목표: 최적의 H(x) 구하기

Output에서 input을 뺀 값인 잔차: F(x) = H(x)-x 를 학습-> Rseidual Learning

목표: F(x)가 0이 되도록

Residual Learning



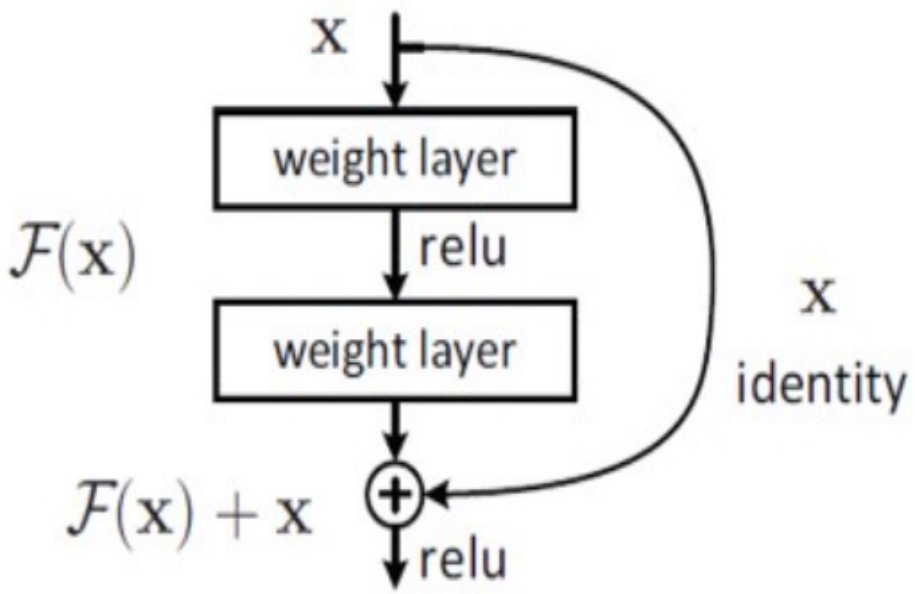
<ResNet>

결과 값에 Input 값 x를 더해주는 것 하나 추가

결과 값에 x를 더한 F(x)+x 가 원래 의도했던 H(x)와 같아지도록 유도 -> 잔차: F(x) = H(x)-x 가 0이 되도록

앞서 학습된 정보 그대로 가져오고 추가적으로 F(x) 만 학습할 수 있어 각각 학습해야 했던 H(x)보다 학습이 훨씬 쉬움

Residual Learning



<ResNet>

$$output = X_{l+1}$$

$$X_{l+1} = f(F(X_l, W_l) + h(X_l))$$

Convolutional layers — Shortcut

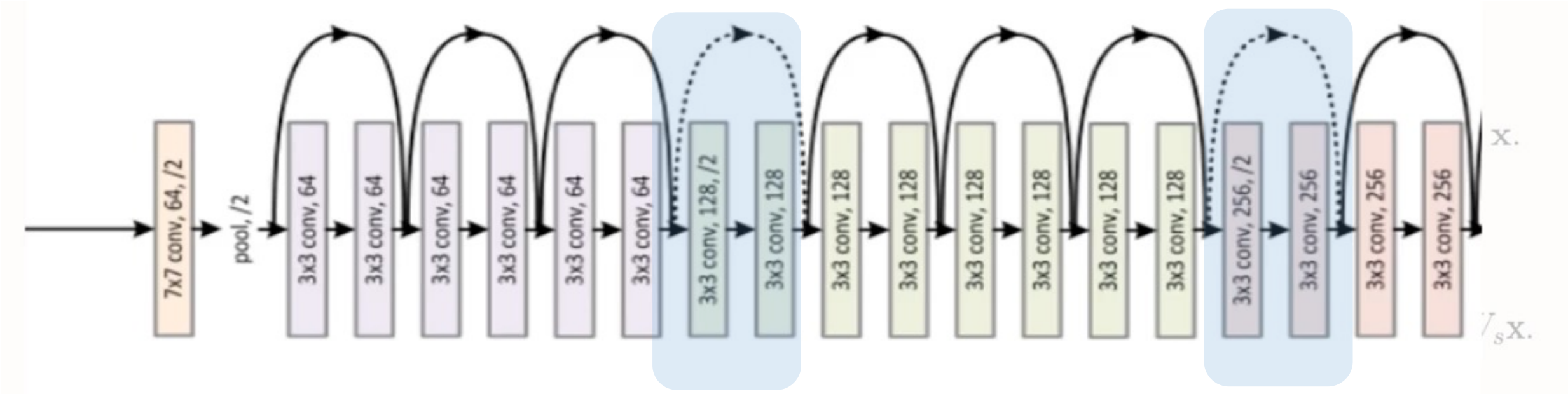
$$X_L = X_l + \sum_{i=1}^{L-1} F(X_i, W_i)$$

$$\frac{\partial \varepsilon}{\partial X_l} = \frac{\partial \varepsilon}{\partial X_L}\frac{\partial X_L}{\partial X_l} = \frac{\partial \varepsilon}{\partial X_L}\left(1 + \frac{\partial}{\partial X_l}\sum_{i=1}^{L-1} F(X_i, W_i)\right)$$

앞의 Gradient가 그대로 더해지면서 Vanishing 문제 해결
(Weight layer를 고려하지 않고 더해지는 형태)

ResNet

"기본적으로 VGG와 비슷, 건너건너 shortcut을 더하는 형태"



실선: Identity Shortcut, 점선: Projection Shortcut

Identity Shortcut VS Projection Shortcut

"Residual 네트워크를 구성할 때 입력과 출력의 차원이 같아야 함"

- ✓ Identity Shortcut: Zero-padding하여 차원을 맞추는 것
- ✓ Projection Shortcut: Linear Projection(1X1 convolution layer 통과) 으로 차원을 맞추는 것

Identity Shortcut의 성능이 떨어지는 이유
: Zero-padding한 영역에서는 residual learning이 되지 않았기 때문

C>B>A> Plain

- (A) : zero-padding(identity) shortcut만 사용
- (B) : 차원이 같은경우 identity, 차원이 다른경우 projection shortcut 사용
- (C) : projection shortcut만 사용

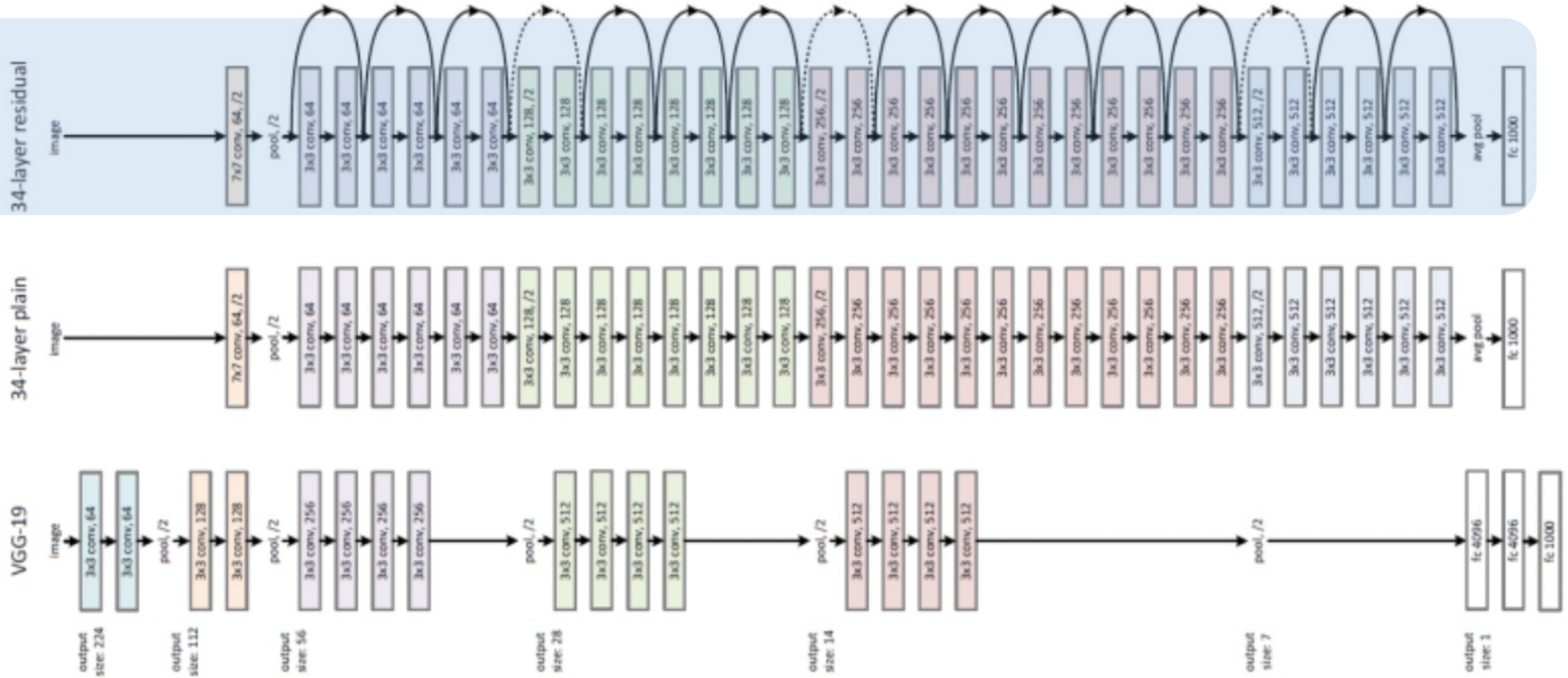
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

ResNet

"3 by 3 convolution을 반복하는 점에서 VGGNet과 비슷"

Residual shortcut 추가

3by 3 convolution을 더 많이 쌓아준 것



ResNet

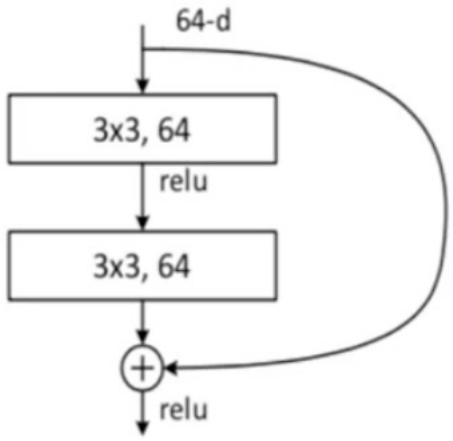
Win

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	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

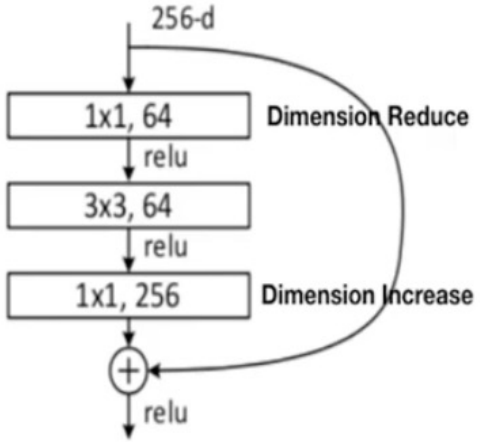
Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet

“ResNet 500이상 모델 - Bottle Neck 구조”



<기본 Residual Block>



<Bottle Neck Residual Block>

1 by 1 필터를 통해 차원 축소
3 by 3 필터 통과
다시 1 by 1 필터를 통해 차원을 키워 줌

-> 피쳐 맵의 개수가 줄어들어 연산 량을 줄여줄 수 있음

Conclusion

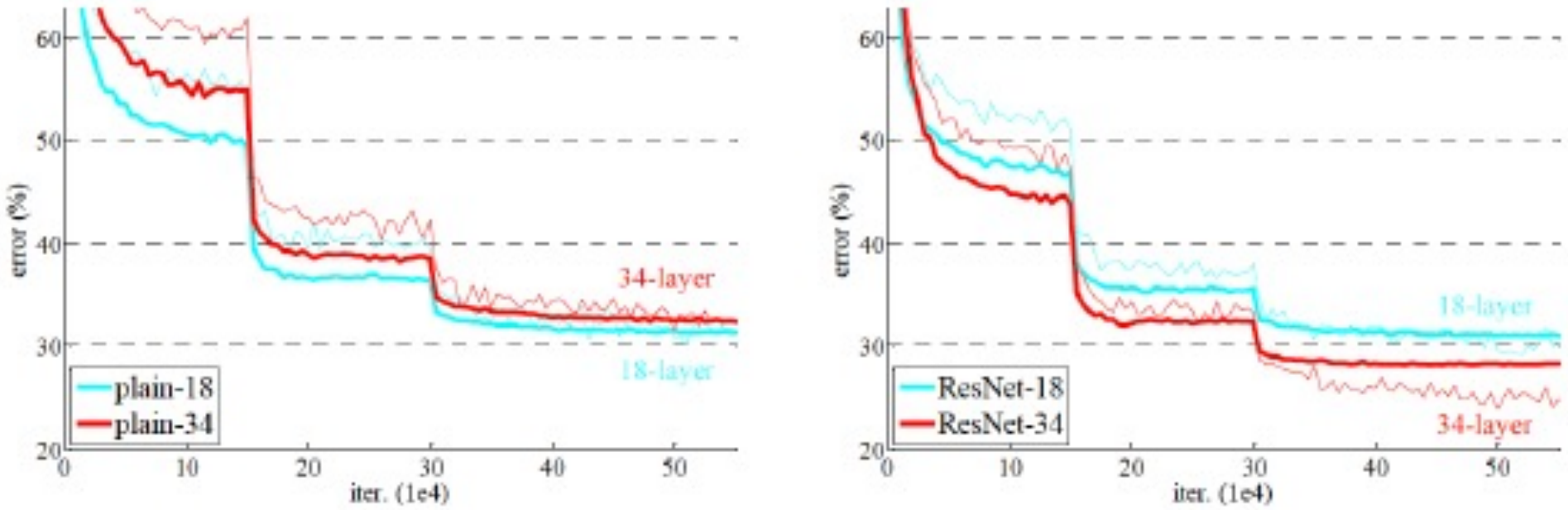


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Conclusion

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

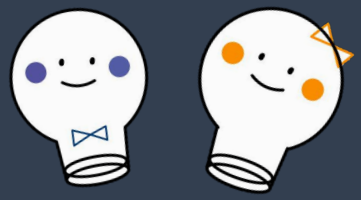
Top-1 validation error rates (%)

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**

Top-5 test error rates (%) of **ensembles**

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Validation error rates (%) of **single-model**



우리만 따라와 Follow ADS

이상 ADS 황성아 였습니다. 감사합니다.