Deep Residual Learning for Image Recognition - ResNet

Abstract

딥러닝에서 neural networks는 깊어질 수록 성능이 train이 어렵다는 것으로 알려져있습니다.

그래서 이 논문은 **residual learning (잔차 학습)** 을 이용해서 더 깊은 신경망에서도 training이 쉽게 이뤄질 수 있다는 것을 제시하였습니다.

We explicitly reformulate the layers as learn-ing residual functions with reference to the layer inputs, in-stead of learning unreferenced functions.
입력 값에 대한 참조를 통해 무참조 함수를 학습하는 대신에 직접 입력값을 참조하여 residual learning 함수로 학습하는 방식으로 재정의 하였습니다.

이 논문은 comprehensive empirical evidence 를 제시하며 residual networks 를 통해 더 쉽게 optimize 하며, 상당히 증가한 깊이로부터 정확도를 얻을 수 있습니다.

ImageNet dataset 에서 최대 152 layers 까지 이루어진 residual nets를 평가하고 VGG nets 보다 8배 더 깊은 신경망을 가지며, 더 낮은 complexity를 가지게 됩니다.

이러한 residual nets의 앙상블은 ImageNet test set에서 3.57% 오류를 가지게 됩니다.

이러한 깊이의 표현은 시각적 인식의 일 중 중요한 일입니다.

이러한 극도의 깊은 표현 덕분에 COCO object detection dataset 에서 상대적으로 28% 상승을 보여주었습니다.

Introduction

Deep convolutional neural networks 는 Image classification에 돌파구를 이끌어왔습니다.

Deep networks는 자연스레 low, mid, high level의 특징들과 분류기를 다층구조로 통합하며, 그 특징들의 수준은 쌓인 레이어의 수로 풍부화 될수 있습니다.

최근 evidence에서 **네트워크의 깊이**가 결과를 이끌어내는데 결정적으로 중요하다고 합니다.
ImageNet dataset의 주요 결과는 very deep 한 모델을 활용하고있습니다.

깊어지는 layer와 함께 떠오는 의문은
더 많은 레이어를 쌓는 것으로 더 좋은 네트워크를 학습하는 것으로 network의 성능이 좋아질까

?

이와 관련하여 Gradient vanishing / exploding과 같은 악명 높은 문제가 발생하였기 때문이다.

그래도 이러한 문제는 normalized initialization, intermediate normalization layers 그리고 수십개의 layers의 SGD 와 같은 다양한 방법들로 개선되어 왔습니다.

이 논문에서 깊게 다룰 문제는 **Degradation Problem** 입니다.

network의 깊이가 깊어질 수록 accuracy는 포화되며 성능이 저하되는 문제입니다.

이는 overfitting에 의한 문제가 아니며 또, 적절한 layer를 추가하면 higher training error를 이끌게 됩니다.

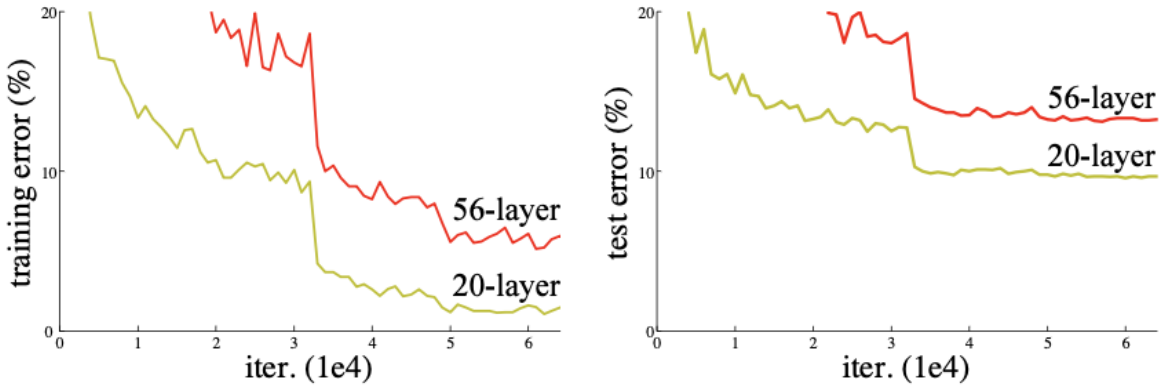

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

fig1. 를 참고하면 56-layer가 20-layer보다 train error, test error 모두 높음을 알 수 있습니다.

degragation 문제는 모든 시스템에 optimize 하기 쉬운 것이 아니라는 뜻이기 때문에 shallower architecture 와 more deep architecture와 비교하려고 한다.
(즉 이 논문에서는 degradation의 문제는 더 깊은 레이어가 쌓일 수록 optimize가 복잡해지기 때문에 일어나는 부작용으로 보고 해결하려고 노력합니다.)

There exists a solution by constrcuction to the deeper model :
학습된 얇은 모델로부터 복사된 다른 layer - identity mapping을 추가해보았지만 실험을 통해 좋은 solution이 아니라는 결과를 얻었습니다.

그래서 이 논문에서는 **Deep residual learning framework** 개념을 도입합니다.

stacked layers에 **desired underlying mapping** 을 직접적으로 하는 것이 아니라 명시적으로 **residual mapping**에 적합하도록 만들었습니다.

desired underlying mapping 은 $H(x)$이면 이 논문에서는 비선형적인 layer의 적합인 $F(X) := H(x) - x$ 를 제시합니다.

이를 전개하면 $H(x) = F(x) + x$ 의 형태가 됩니다.

여기서는 residual mapping이 기존의 mapping 보다 optimize 하기 쉽다는 가설을 세웁니다.

extreme 하게 identity mapping이 최적이라고 할 때, 비선형 레이어를 stack 하여 identity mapping을 맞추는 것 보다, residual mapping을 0으로 만드는 것이 더 쉽습니다.

$F(x) + x$ 는 “Shortcut Connections”와 동일한데
Shortcut Connection은 1개의 layer 혹은 더 많은 layer를 skip 하게 만들어줍니다.

이 논문에서는 identity mapping으로 shortcut connection이 단순히 수행되게 한다.

Identity Short Connection은 **추가의 파라미터도 필요하지 않고**, **복잡한 계산 또한 필요하지 않은 것**이 장점이다.

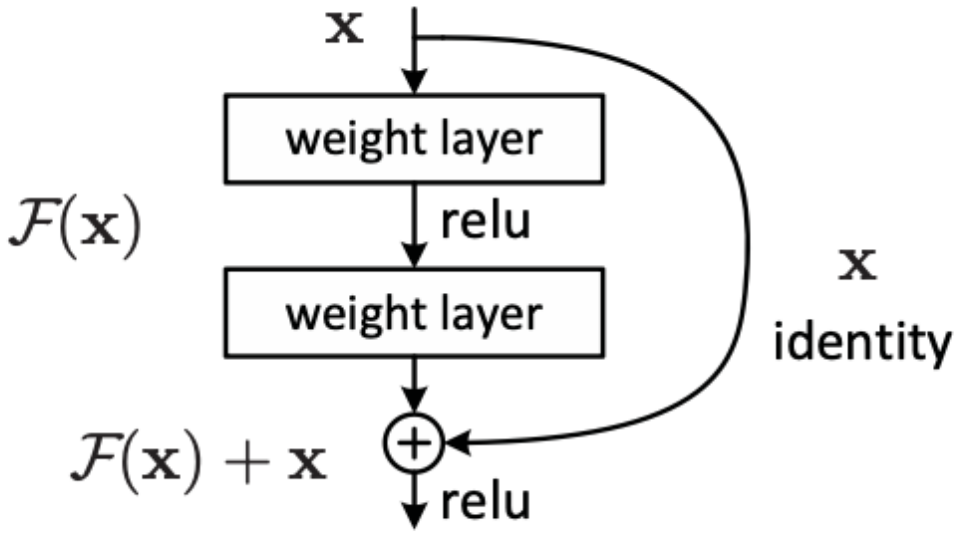

Figure 2. Residual learning: a building block.

x 는 input으로 Model 인 $F(x)$를 거치면 자신(identity)인 x가 더해져서 output으로 $F(x) + x$ 가 나오는 구조입니다.

($F(x)$ 는 model, relu 는 fuction)

이 논문의 목표 두 가지는 다음과 같습니다.

1) Our extremely deep residual nets are easy to optimize, but counterpart “plain” nets (that simply stack layers) exhibit higher training error when the depth increases;

plain net과 다르게 residual net이 더 낮은 오류율을 보이며 더 쉽게 최적화 되는 것을 보이기

2) Our deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks

Deep residual net가 이전의 network 보다 더 쉽게 증가한 깊이에서 accuracy를 얻을 수 있다.

Fisher Vector는 컴퓨터 비전 및 패턴 인식 분야에서 사용되는 특징 벡터 표현 방법입니다. 이 방법은 확률적 모델을 사용하여 특징 벡터의 통계적 특성을 모델링하고, 벡터의 각 성분이 얼마나 중요한 정보를 담고 있는지를 평가합니다. Fisher Vector는 일반적으로 이미지 분류, 객체 인식 및 검색 등의 작업에서 특징 벡터를 효과적으로 표현하는 데 사용됩니다.

Related Work

Residual Representations

Image recognition 에서 VLAD는 사전에 대한 residual vector 를 인코딩 하는 표현 방법이며, Fisher Vector는 VLAD의 확률적 버전으로 수식화 될 수 있습니다.

Vector quantization에서는 residual vector를 인코딩 하는 것이 원래 벡터를 인코딩하는 것 보다 효과적임을 입증되었습니다.

PDE는 편미분 방정식을 해결하기 위해 널리 사용되는 Multigrid 방법으로 시스템을 여러 스케일의 하위 문제로 정의합니다.

각 subproblem은 coarser , finer scale의 residual solution에 대해 responsible 합니다.

Shorcut Connections

몇몇의 intermediate layers 들은 gradient vanishing / exploding을 다루기 위해 보조의 분류기를 직접적으로 연결합니다.

구체적으로 highway networks 는 gating 함수를 사용한 단축 연결을 제시하였으며 이러한 게이트들은 data에 의존적이며, 많은 parameters 를 가지고 있으며, 우리의 매개변수 없는 identity skip connection 과는 대조적입니다.

gate shortcut이 closed 되어 있을 때, highway networks는 non-residual function 을 나타내지만 대조적으로 residual function은 항상 학습을 하기에 identity shortcut은 절대 닫히지 않고, 모든 정보는 항상 통과하며, 추가적인 residual 함수는 학습을 할 것 입니다.

또한, highway network는 극단적인 깊이의 증가와 정확도 향상을 보이지 않았고.

즉, 깊이가 극단적으로 증가할 때 high way network는 성능 향상을 제한하는 경향이 있습니다.

Deep Residual Learning

Residual Learning

$H(x)$ 를 기존의 네트워크라고 할 때 이 $H(x)$는 여러 비선형 layer로 이루어져 천천히 복잡한 함수에 근사된다고 가정 할 때, $F(x) := H(x) - x$로 바꿔 $F(x) + x$ 를 $H(x)$ 에 근사하도록 하는 것(Residual mapping)이 더 쉽다고 가정합니다.

이를 feed-forward neural network에 적용한 것이 Shortcut connection 입니다.

Identity mapping by Shortcuts

Residual Learning 의 중점은 Degradation 문제를 어떻게 해결하느냐 인데, 위(**Fig.2**) 에 나왔던 거 처럼 $F(x) + x$ 를 $H(x)$ 에 근사하게 한다. shorcut connection을 Neural Network에 사용하여 back propagation 할 때, identity mapping 을 미분하면 적어도 1 이상의 값이 나와 최소한의 기울기를 만들어서 학습이 제대로 일어나지 않는 현상을 최소화 했습니다.

$H(x)$ 와 $F(x) + x$ 가 천천히 근사하는 방식은 같더라도 학습의 속도나 안정성을 보았을 때 Shortcut connection이 훨씬 더 효율적이다.

$$y = F(x, W_i) + x \text{ , } F = W_2\sigma(W_1x) \qquad (1)$$

기본적으로 residual block을 다음 수식과 같이 정의한다. 이때 σ 는 ReLu 이고, bias는 표기상 생략 되었다.

위 경우에서는 x와 F의 dimension이 동일해야하며, 만약 dimension이 바뀔 경우 Linear projection W_s 를 적용할 수 있다.

여기서 W_s 는 dimension을 맞추기 위해서만 사용된다.

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

여기서 (2) 를 projection shortcut connection이라고 부른다.

Network Architectures

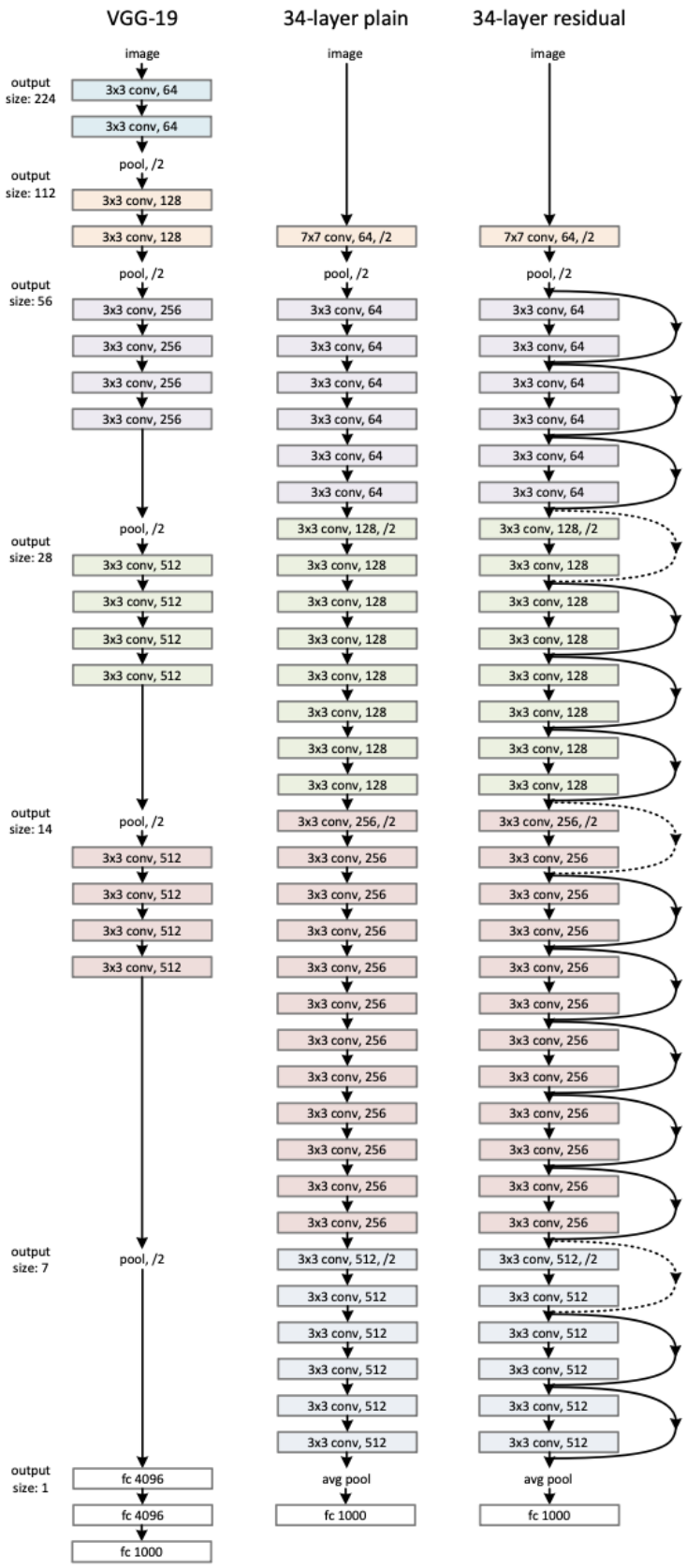
VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Plain Network

Residual block을 사용하지 않고, 단순히 쌓은 네트워크(**Fig. 3, middle**)입니다.
이 네트워크는 VGG(**Fig. 3, left**) 에서 영감을 받아 만들었습니다.

plain network 는 두가지 design rule이 있습니다.

1. 각 layer 들은 같은 크기의 feature map output size를 만들고, 동일한 numer of filters를 가집니다.
2. feature map size 가 반으로 줄어들 경우, layer당 complexity를 유지하기 위해 number of filters를 2배로 늘립니다. convolution layer의 stride를 2로 하여 down-sampling 을 직접 수행합니다.

convolution layer(34개)를 통과한 뒤, global average pooling, 1000-way fully-connected layer with softmax 로 끝납니다. 또한 이 모델은 VGG Net 보다 fewer filters, lower complexity를 가지고 있고, layer baseline은 3.6billion FLOPs로서 VGG의 18% 입니다.

Residual Network

plain network에 base를 두고 skip connection을 추가한 ResNet(**Fig. 3, right**) 입니다.

shortcut connection(**Fig. 3, solid line**) 은 input과 output이 같은 dimension일 때 사용됩니다. dimension이 증가할 때 (**Fig. 3, solid line**) 는 두 가지 경우를 고려하였습니다.

1. identity shortcut connection 을 계속 실행하는 경우, dimension을 증가시키기 위해 나머지를 zero padding 을 하였습니다.
2. projection shortcut connection은 dimension을 맞추기 위하여 1x1 convolution을 사용합니다.

양 쪽 옵션 모두 stride 는 2를 사용합니다.

Implementation

- weight decay 0.0001 , Momentum 0.9
- Dropout X
- image 는 [256, 480] 중 가까운 방향으로 resize

- per-pixel mean subtraction 과 함계 224x224 로 랜덤 crop
- standard color augmentation 사용
- Batch Normalization 을 convolution 후, activation 전 사용
- batch size 256으로 SGD 사용
- Learning rate 는 0.1부터 시작, local minimum을 만나거나 loss가 진동하는 경우 1/10씩 감소시킴
- iteration : $60 * 10^4$

테스트시 대회를 위해 standard 10 crop testing을 적용하였고, fully convolutional을 적용하였습니다.

Experiment

ImageNet Classification

ImageNet classification dataset에서 model을 평가하였고, 학습이미지는 1.28백만장, 평가는 5만장을 이용, 그리고 서버로 보내는 최종 테스트시에는 10만장을 이용하였고, top -1 top -5 에러율을 평가하였습니다.

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

Table 1. Architecture for ImageNet. Building blocks are shown in brackets, with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4-1, and conv5_1 with a stride of 2.

위 그림은 ImageNet 참가시 사용한 Network Architecture Implementation detail 입니다.

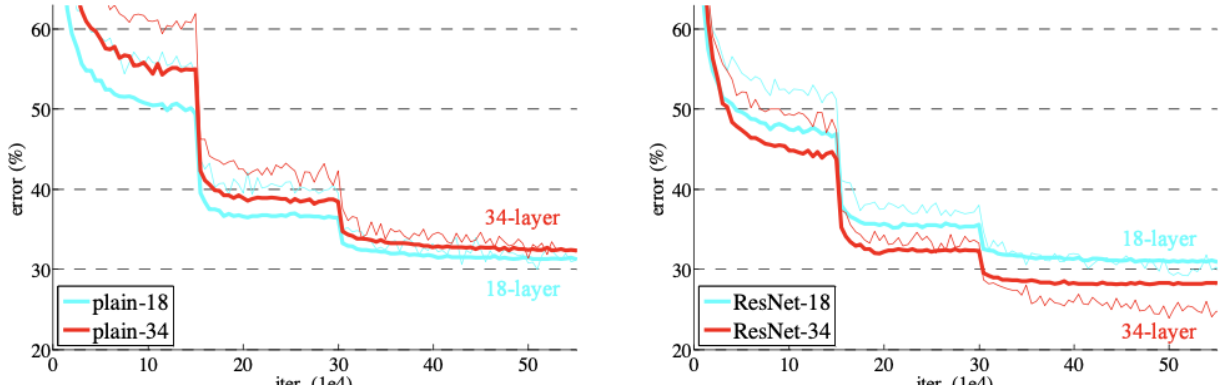

Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Fig. 4, left를 보면 plain-18 이 34보다 전체적인 error가 적은 것을 볼 수 있습니다.

	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.

Plain Network

Plain Net으로 쌓은 18 -layer 와 34-layer를 비교하면 34-layer에서 train error, test error 모두 높은 degradation 문제를 볼 수 있다.

plain net이 Batch Normalization으로 학습 되었기 때문에 non-zero varience 를 가지고 optimization 이 어려운 이유가 vanishing gradient 때문이 아닌 것을 알 수 있다.

→ 기하급수적으로 수렴이 어려워지는 것을 training error의 reducing 에서 어려운 점으로 본다.

Residual Network

ResNet-18, 34 모두 plain network에 base를 두고 있고, 첫번째 비교(Table 2 and Fig. 4) 에서 identity shorcut conecction을 사용하였습니다(zero padding). 그리고 Table2와 Fig. 4로부터 3가지 요점을 찾을 수 있습니다.

1. ResNet 을 사용할 경우 Network의 depth가 증가하여도 error가 감소한다. (degradation 문제 해결) 같은 depth 여도 ResNet 성능이 더 좋습니다(Table2).
2. plain 보다 ResNet이 더빨리 solution을 찾아갑니다. (Fig. 4)
3. 18-layers 들 끼리 비교했을 때 accuracy는 비슷했지만 19-layer ResNet이 더 수렴이 빨랐습니다.

→ ResNet이 SGD를 사용한 optimization이 더 쉬움을 알 수 있다.

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.

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.

Identity vs. Projection Shortcuts

training 에서 identity shortcuts 를 돕기 위해서 parameter-free 방법을 사용하는데 projection shortcut과 비교해보겠습니다.

A. 차원 증가를 위해 zero-padding을 사용하는 경우 (추가 파라미터 없음)

B. 차원 증가를 위해 projection shortcut을 사용, W_s 다른 shortcuts는 identity

C. 모든 shortcut 들이 projection인 경우

A는 residual learning이 이루어지지 않기 때문에 B가 성능이 미세하게 더 좋고, C가 B보다 좋은 이유는 projection shortcut에 사용된 extra parameter 때문입니다.

C가 성능이 제일 좋지만 많은 parameter들이 추가되어 mem/time 이 복잡하고 모델 사이즈 때문에 이 논문에서는 사용하지 않는다.

A/B/C 간의 차이는 Projection shortcut 경로가 저하문제를 해결하는데 필수적이지 않습니다.

Identity Shortcuts는 bottleneck architecture 의 복잡성을 막기 위해 특히나 중요하다.

Deep Bottleneck Architecture

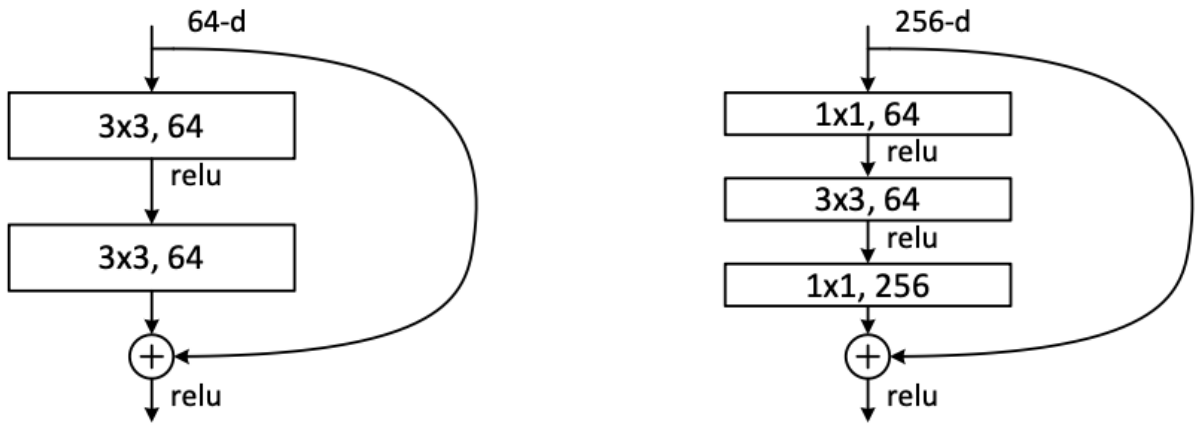

Figure 5. A deeper residual function F for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet- 34. Right: a “bottleneck” building block for ResNet-50/101/152.

train 시간을 고려해 아키텍쳐의 구조를 Bottleneck design을 사용합니다.

각각 Residual function F를 사용하는데 3층을 쌓을 때, (1 x 1) → (3 x 3) → (1 x 1) 순서로 병목 모양으로 쌓습니다.

→ 이는 기존 구조와 비슷한 복잡성을 가지면서 input 과 output의 dimension을 줄이기 때문에 사용합니다.

50-layers

기존에 만들었던 34-layer에다가 3-layer Bottleneck block을 추가해서 50-layer ResNet 을 만듭니다

옵션 B 사용, 연산은 3.9 billion FLOPs

101-layer and 152-layer ResNets

더 많은 3-layer을 추가해서 101-layer와 152-layer를 만듭니다.

이중 152-layer ResNet이 VGG보다 작은 복잡성과 적은 연산을 가져서 유의미 합니다.

또한 50/101/151 layer는 34-layer 보다확실히 더 정확합니다. (degradation problem x, 깊이의 이점을 살림)

Comparisons with State-of-the-art Methods

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**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Table4 를 참고하면 결과적으로 각기 다른 깊이의 6가지 모델을 ensemble error을 3.57% 까지 줄였습니다.

CIFAR-10 and Anaysis

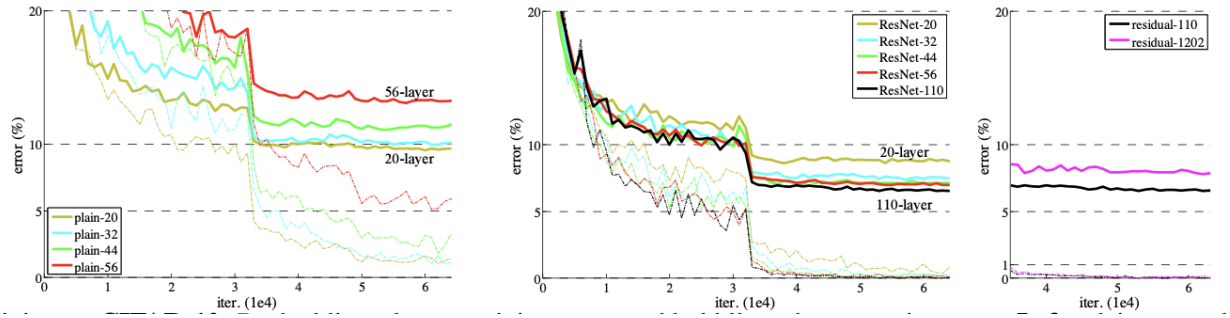

Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

CIFAR-10 test set을 이용하였을 때 각 layers 별로 성능 체크를 하였고, layer가 너무 깊을 경우 (**residual 1202**) 다시 error가 증가하는 것을 볼 수 있습니다

그 이유는 overfitting 현상이고, 이 논문에서 dropout, maxout 등의 regularization이 쓰이지 않기 때문에 추후 regularization으로 이 문제를 해결해야 합니다.

Fig. 6 는 CIFAR-10 에서 학습을 진행할 때 plain network 같은 경우에는 깊을 수록 학습이 잘 진행이 되지 않았지만 ResNet의 경우에는 반대의 양상을 띄게 됩니다. (ResNet-1202는 제외)

Analysis of Layer Responses

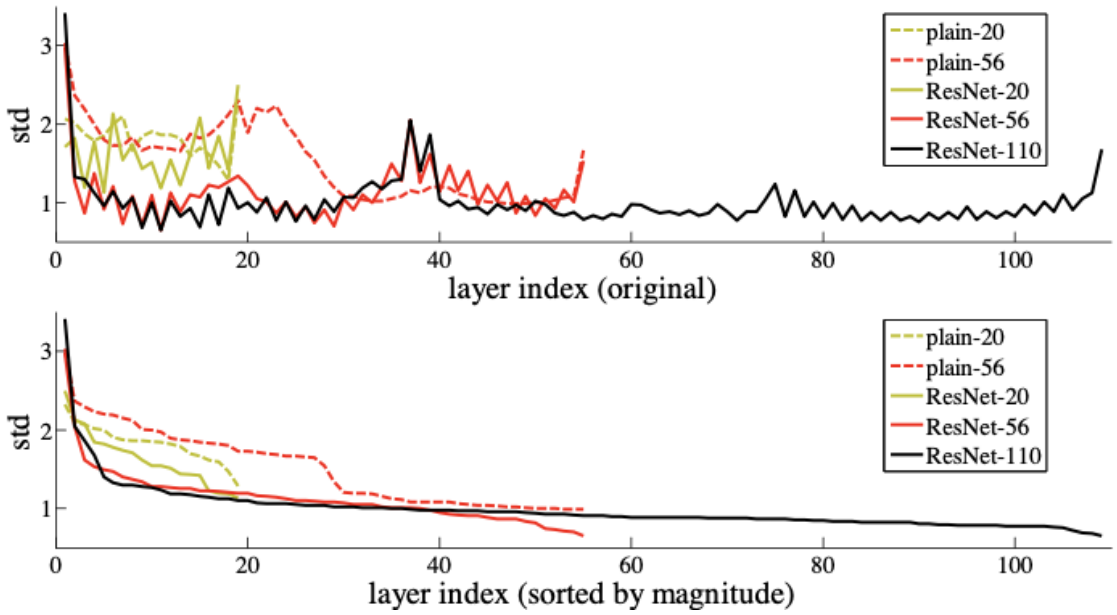

Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

각 3x3 convolution layer (Batch Norm 후, ReLu 전) 의 표준 편차입니다.

ResNet이 plain network 에 비해 small response 를 보이는 것으로 보아, 좀 더 안정적으로 학습을 진행한다고 할 수 있습니다.