

# Deep Residual Learning for Image Recognition

13기 이유민

## 0. Abstract

## 1. Introduction

## 2. Related Work

## 3. Deep Residual Learning

- 3.1. Residual Learning
- 3.2. Identity Mapping by Shortcuts
- 3.3. Network Architectures
- 3.4. Implementation

## 4. Experiments

- 4.1. ImageNet Classification
  - 1) Plain Network
  - 2) Residual Network
  - 3) Identity vs. Projectoin Shortcuts
  - 4) Deeper Bottlenec Architectures
  - 5) 50-layers ResNet
  - 6) 101-layers ResNet
  - 7) Comparisons with State-of-the-art Methods
- 4.2. CIFAR-10 and Analysis
- 4.3. Object Detection on PASCAL and MS COCO

## 5. References

## 0. Abstract

이 논문은 Residual Learning Framework를 통해 깊은 Neural Network(NN)을 좀 더 쉽게 학습할 수 있는 방법을 소개하고 있다.

기존의 방법과는 다른 점은 레이어에서 받은 입력을 참고해서 얻은 잔차(Residual)를 학습한다는 점이다.

논문에서는 소개한 방법(ResNet)을 통해 더 높은 정확도를 얻을 수 있고 최적화하기도 쉽다는 걸 증명했다.

- ImageNet dataset에서 에러 3.57% (VGG nets보다 8배 깊지만 복잡도는 낮음)
- ILSVRC 2015 classification task 1위
- CIFAR-10 with 100 and 1000 layers에서의 analysis 도출 가능
- tasks of COCO object detection dataset : 28% 개선
- ImageNet 감지, ImageNet 현지화, COCO detection, COCO segmentation

# 1. Introduction

- Depth 를 늘리면 성능이 좋아질까 (x)

**degradation** : depth 가 깊어질수록 성능이 좋아지지가 특정 부분에서부터 다시 성능 ↓ 현상

: why? ) 그라디언트가 vanishing & exploding 하기 때문!

↳ 원인 ) overfitting (x) .

training error ↑ (o)

- 어떻게 해결할까?

: Let 기본의 네트워크 :  $H(x)$  ,

$F(x)$  : 잔차 ,  $x$  : 원래 input

$$\Rightarrow H(x) = F(x) + x$$

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

layer 를 그대로 통과

- \* **Shortcut Connection** ( $\Leftrightarrow$  main connection)

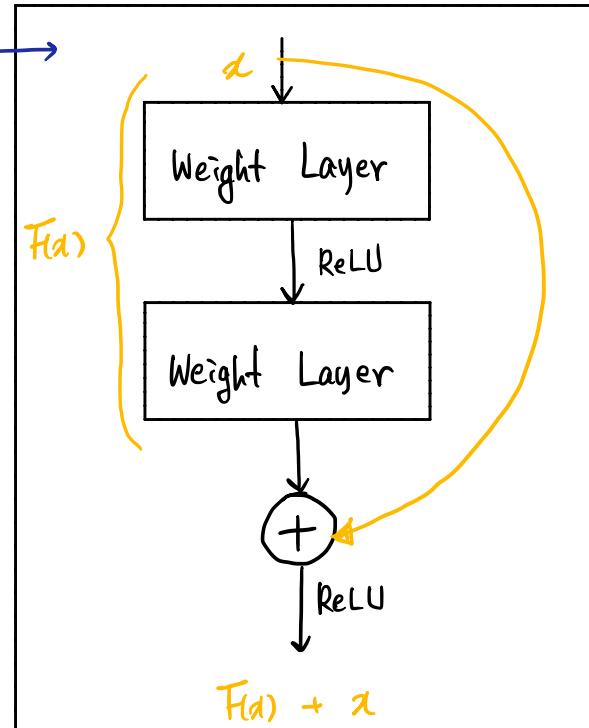
: Layer 를 건너뛰는 것

: direct mapping (x), identity mapping (o)

$$\Rightarrow H(x) \text{ 대신 } F(x) + x \text{ ↳ identity mapping!}$$

장점) 계산이 많이 복잡해지지 x, parameter 추가 x

↳ SGD 와 Backpropagation 으로 전체 네트워크 학습 o!



< figure 2 >

shortcut connection 과  
main connection 의  
결과를 더해서  
activation 하는 것!

- 更深 평가 & Degradation 문제 해결 방법은?

: ImageNet 으로 보이자!

1) [ ResNet ] : 깊어져도 최적화 easy

[ Plain Network ] : 깊어질수록 training error 증가

2) [ ResNet ] : 깊어져도 accuracy 도출 easy

[ Plain Network ] : ResNet 보다 accuracy 도출 어려워함

↳ ResNet 은 극단적으로 깊어져도 일반화하는 데에 성능 good !

## 2. Related Work

1. Residual Representations
2. Shortcut Connections

## 3. Deep Residual Learning

### 3. 1 Residual Learning

· 가정) 비선형 layer들이 복잡한 함수 ( $H(x)$ )에 근사하는 경우

→  $F(x) + x$  를  $H(x)$ 에 근사하는 게 better : Reformulation

But, 더 깊은 모델의 training error가 더 낮을 수 있다는 상식에 반함.  
(by 1. Introduction)

· multiple nonlinear layer가 복잡한 함수를 근사할 수 있는 경우,  
Residual function이 근사 가능!

### 3. 2 Identity Mapping by Shortcuts

· ResNet은 Residual Learning을 이용한다는 점을 기억하고 보자.

i) Let  $x$ : input  $\vec{x}$ ,  $y$ : output (벡터 형식)

by <figure 2>. Let  $y = \underbrace{F(x, W_i)}_{\text{앞으로 학습되었던}} + x \dots \text{식 (1)}$   
앞으로 학습되었던 residual mapping 값!

ii) figure 2를 수식으로 나타내자.

$F = W_2 \sigma(W_1 x)$  where  $\sigma$ : ReLU, 전의상 bias 생략

\* Input / output의 차원 (channel)이 다를 경우 차원을 같게 만들어줘야 함.

↳  $y = F(x, W_i) + \underbrace{W_s x}_{\text{only 차원 맞추는 역할 수행}} \dots \text{식 (2)}$

+ skip은 2개 이상씩 해야 함 (stride  $\geq 2$ )

Why?) 하루에 skip하면  $y = Wx + x$  (Linear Layer) → 희석 효과가 ×

### 3.3 Network Architectures

- Plain Network vs. Residual Network by ImageNet data

\* Plain

Layer 쌓기 '만'

- ① input & output feature map 크기  $\ominus$ , filter 수  $\ominus$
- ② if featuremap size  $\downarrow$  (이 경우는 절반)  $\Rightarrow$  for 시간복잡도 유지  
 $\hookrightarrow$  filter size  $\uparrow$  (.. 2배)
- ③ convolution Layer 중 stride = 2 인 경우  
: downsampling 역할
- ④ VGG<sub>16</sub>에 비해 ① filter 수  $\downarrow$ , ② 복잡도  $\downarrow$   
 $\downarrow$  net

\* Residual

Plain Network의 Shortcut Connection의 추가된 형태

- ① if input 차원 = output 차원  $\Rightarrow$  바로 shortcut 적용 가능

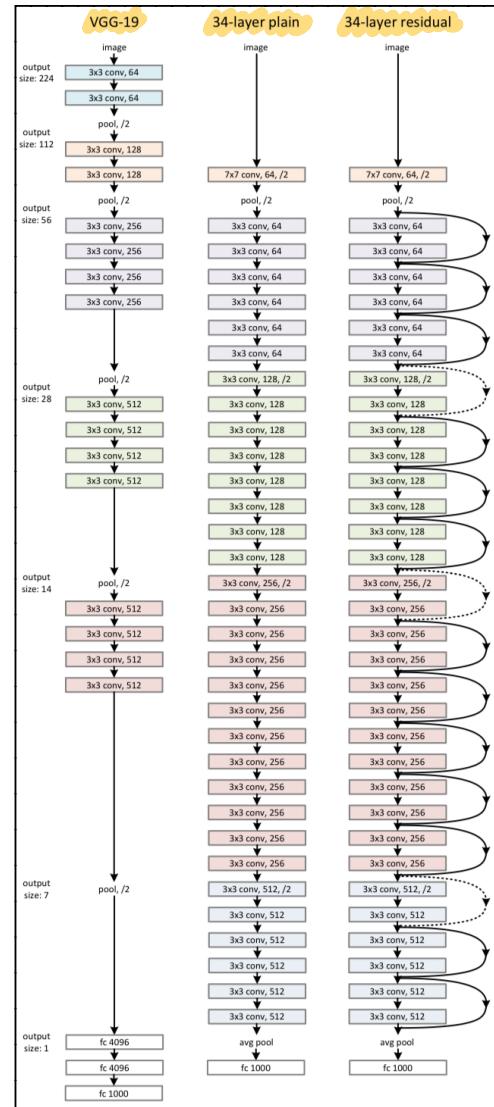
- ② if " < "

$\Rightarrow$  Sol 1) zero padding (shortcut으로 identity mapping은 해제)  
 $\hookrightarrow$  extra parameter가 필요없는 이유

$\Rightarrow$  Sol 2) 수식 2에서 shortcut 뒤에 1x1 Convolution 사용

### 3.4 Implementation

- Imagenet data에 적용시켜 보자.
- : [2tb, 480] 중 random sampling.
- : standard color augmentation 사용
- : convolution 전, activation 후에는 Batch Normalization
- : mini batch 256, SGD use
- : test는 10 - crop testing  
이미지를 10번 잘라서 봄
- 초기값 : plain network에서 학습한 걸 갖다쓰자



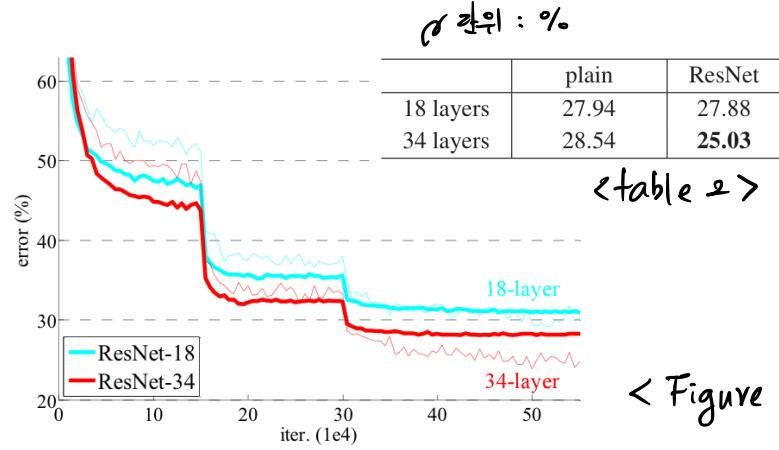
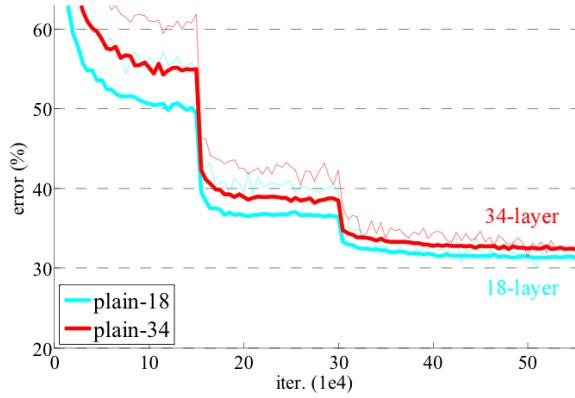
# 4. Experiments

## 4.1 ImageNet Classification

<table 1>

- 설명 dataset
- 1) ImageNet 2012 classification (1000 class)
- 2) training image : 128만장
- 3) validation image : 50만장
- 4) test image : 10만장
- ⇒ top1 브터 top5 까지 error rate

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$[3 \times 3, 64] \times 2$ $[3 \times 3, 64]$	$[3 \times 3, 64] \times 3$ $[3 \times 3, 64]$	$[1 \times 1, 64] \times 3$ $[3 \times 3, 64]$ $[1 \times 1, 256]$	$[1 \times 1, 64] \times 3$ $[3 \times 3, 64]$ $[1 \times 1, 256]$	$[1 \times 1, 64] \times 3$ $[3 \times 3, 64]$ $[1 \times 1, 256]$
conv3_x	28×28	$[3 \times 3, 128] \times 2$ $[3 \times 3, 128]$	$[3 \times 3, 128] \times 4$ $[3 \times 3, 128]$	$[1 \times 1, 128] \times 4$ $[3 \times 3, 128]$ $[1 \times 1, 512]$	$[1 \times 1, 128] \times 4$ $[3 \times 3, 128]$ $[1 \times 1, 512]$	$[1 \times 1, 128] \times 8$ $[3 \times 3, 128]$ $[1 \times 1, 512]$
conv4_x	14×14	$[3 \times 3, 256] \times 2$ $[3 \times 3, 256]$	$[3 \times 3, 256] \times 6$ $[3 \times 3, 256]$	$[1 \times 1, 256] \times 6$ $[3 \times 3, 256]$ $[1 \times 1, 1024]$	$[1 \times 1, 256] \times 23$ $[3 \times 3, 256]$ $[1 \times 1, 1024]$	$[1 \times 1, 256] \times 36$ $[3 \times 3, 256]$ $[1 \times 1, 1024]$
conv5_x	7×7	$[3 \times 3, 512] \times 2$ $[3 \times 3, 512]$	$[3 \times 3, 512] \times 3$ $[3 \times 3, 512]$	$[1 \times 1, 512] \times 3$ $[3 \times 3, 512]$ $[1 \times 1, 2048]$	$[1 \times 1, 512] \times 3$ $[3 \times 3, 512]$ $[1 \times 1, 2048]$	$[1 \times 1, 512] \times 3$ $[3 \times 3, 512]$ $[1 \times 1, 2048]$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$



<Figure 4>

### \* Plain Networks

- 1) 34 layers 의 Validation error 가 더 ↑
- 2) " training error 가 더 ↑
- ⇒ But, vanishing gradient의 문제와 더불어 convergence rate가 더욱 작기 때문 (주장)

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	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	<b>21.43</b>	<b>5.71</b>

### \* Residual Networks

- 1) 34 layers 의 Validation error 가 더 ↓
- 2) " training error 가 더 ↓
- ⇒ 같은 system의 경우에는 Residual을 학습하는게 better!
- + optimization 속도도 good.

### \* Identity vs. Projection Shortcuts

- <Table 3> 을 통해 비교할 수 있는 옵션들은 다음과 같음
- A) 차원 증가를 위해 zero padding shortcut 사용 & 모든 short cuts는 parameter free
- B) " Projection shortcut 사용 & 서로 다른 short cuts는 identity 항
- C) 모든 short cuts는 projection 일



<Table 3>

⇒ 세 옵션을 비교해 보자.

우선 plain 보다 residual이 더 좋은 결과를 냄!

⇒ 성능 : C > B > A

① extra parameter 가 많은 projection shortcuts에 의한 것

② A의 zero padding 또는 Residual Learning이 없음

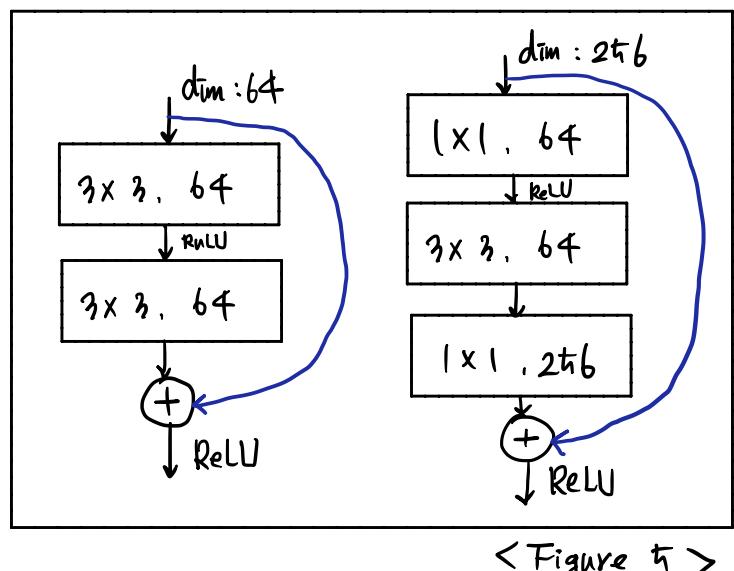
But A, B, C의 성능 차이가 작음

⇒ degradation 해결에 projection shortcut이 필수적이지는(x)

⇒ C는 쓰지 말자! (Memory ↑)

## \* (Deeper) Bottleneck Architectures

- 3 layers에는 Bottleneck 有
- 50-layer 이상이면 → 구조변경
- 차원 축소다가 다시 늘림!
- identity shortcut
  - : 시간복잡도와 모델 크기를 줄이는 데 중요.
- ⇒ bottleneck을 더 효율적으로 디자인할 수 있게 해 줌!



## \* 50-layers ResNet

3 layers의 Bottleneck Block을 50 layers, (B) 옵션으로 차운 ↑

## \* 101-layers ResNet and 102-layers ResNet

50-layers ResNet에서 깊이(depth)에만 변화를 줌.

⇒ depth 차질우록 성능 좋아짐

## \* Comparison with state-of-the-art Methods

method	top-5 err. (test)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	3.57

⇒ <table 5> 와 같이 모델들을 Ensemble하면 error가 only 3.51%.

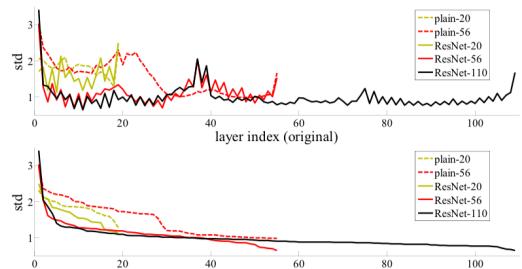
⇒ ResNet의 깊이를 다양하게 하여 Ensemble 해주면 성능 ↑

<table 5 >

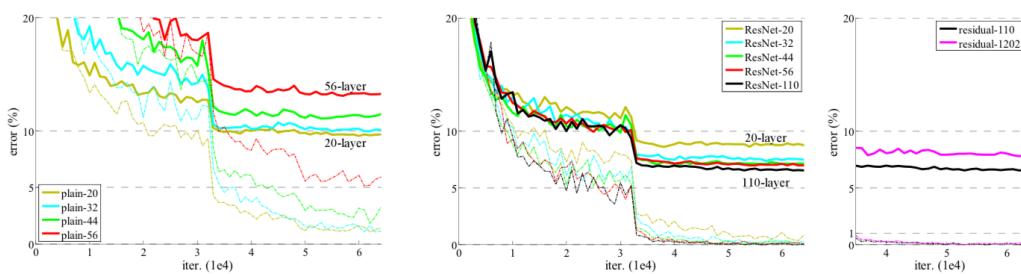
## 4.2 CIFAR-10 and Analysis

### \* Analysis of Layer Responses

plain vs. ResNet의 degradation 정도 & ResNet의 해결策 (o)



### \* Exploring Over 1000 Layers



<Figure 6>

<Table 6>

110-layer vs. 1202-layer 비교했을 때, 1202-layer의 test error ↑.

why?) data 수가 적은데 너무 깊어서 overfitting.

+ maxout, dropout 등으로 해결 가능해 보이긴 함

method	error (%)
Maxout [9]	9.38
NIN [25]	8.81
DSN [24]	8.22
# layers	# params
FitNet [34]	19
Highway [41, 42]	19
Highway [41, 42]	32
ResNet	20
ResNet	32
ResNet	44
ResNet	56
ResNet	110
ResNet	1202

## 4.3 Object Detection on PASCAL and MSCOCO

### recognition tasks

6% more ↑

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>

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	<b>48.4</b>	<b>27.2</b>

<table 8>

## 5. Reference