세 션 9 주 차

ToBig's 12기 이세윤

ResNet

Deep Residual Learning for Image Recognition

(Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun)

Contents

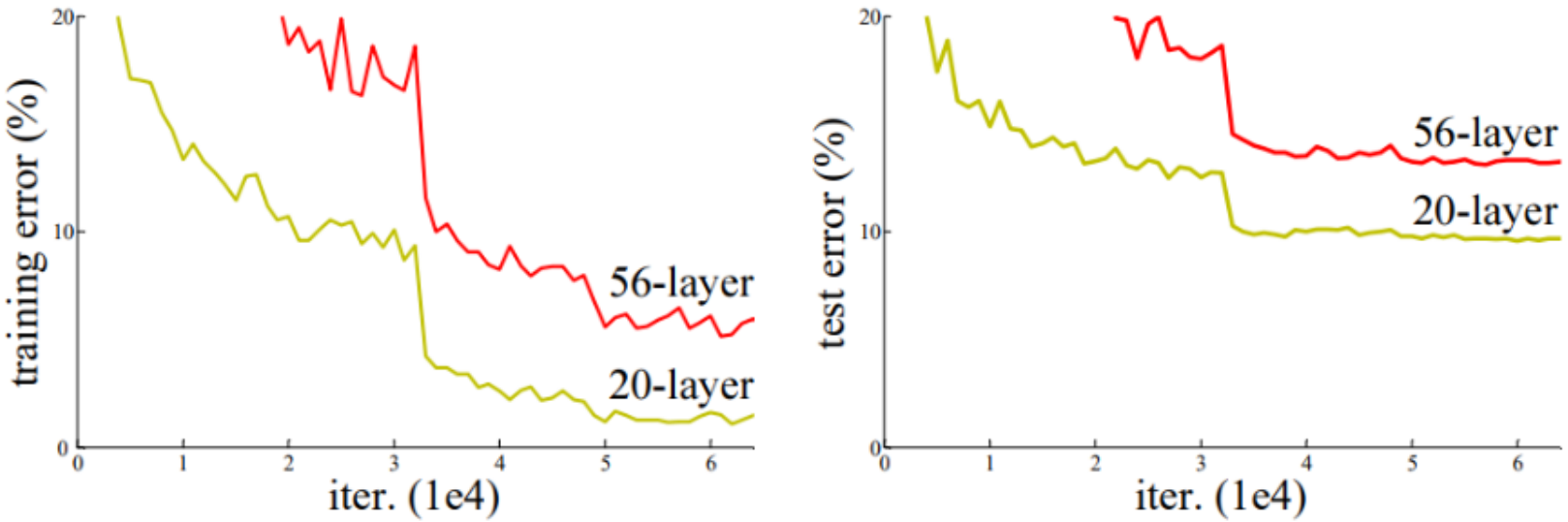
Contents

Unit 01 | Abstract

ResNet 요약

VGGNet에서의 이슈가 있었는데,

바로 CNN에서 층이 너무 깊으면 성능이 오히려 나빠진다는 것!



Unit 01 | Abstract

ResNet 요약

이에 등장한 ResNet은
Residual(잔차)를 학습하는 방법으로 더 깊은 층을 쌓고 학습시키기 용이하도록 하였다.

이 방법은 VGGNet보다 8배 깊은 152개의 층으로 3.57%라는 매우 작은 error을 보이며
2015 ILSVRC에서 1등을 하였다.

Contents

Unit 02 | Introduction

"과연 층을 깊게 쌓는 것만으로 쉽게 성능을 높일 수 있을까?"

깊은 층을 쌓는 실험에서 성능 저하 문제가 일어났다!
- 층의 깊이가 증가하면서 정확도가 ' 포화상태 ' 에 도달하게 되어 성능이 떨어지는 현상

이는 overfitting으로 인한 것은 아니였다.
(overfitting이라면 train에서의 성능은 더 좋아야 하는데 그렇지 않은 것을 알 수 있음)

그렇다면 무엇?
- Degradation 문제!

  저자들은 이 문제가 network가 깊어질수록 vanishing gradient가 크게 영향을 주기 때문이라고 보았다.

아까 봤던 이 그림

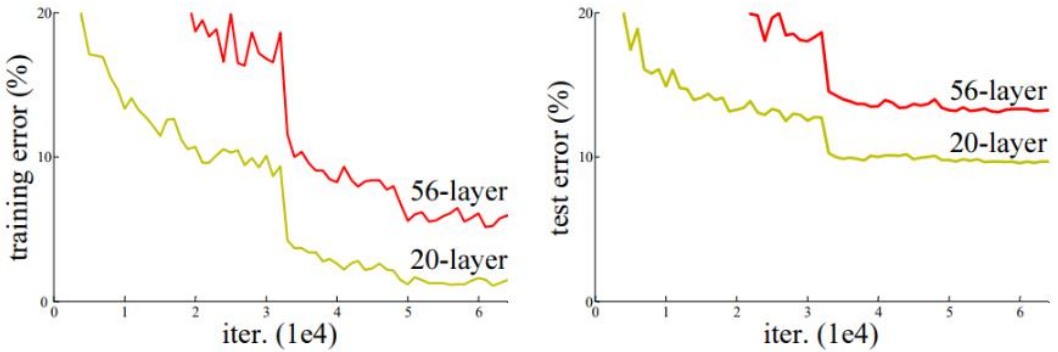


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

20-layer와 56-layer의 network가 CIFAR-10 데이터셋을 학습하였을 때의 결과

층이 더 깊을수록 training error가 더 높았으며, test error 또한 더 높았다.

이유는, **망이 깊을수록 최적화의 난이도가 올라가기 때문!**

Residual Learning Framework을 통해 성능 저하 문제를 해결하자!

Shallow architecture 에 Identity Mapping을 하는 layer만 추가하여 Deeper architecture를 만든다면 성능이 더 나빠지지 않지 않을까?

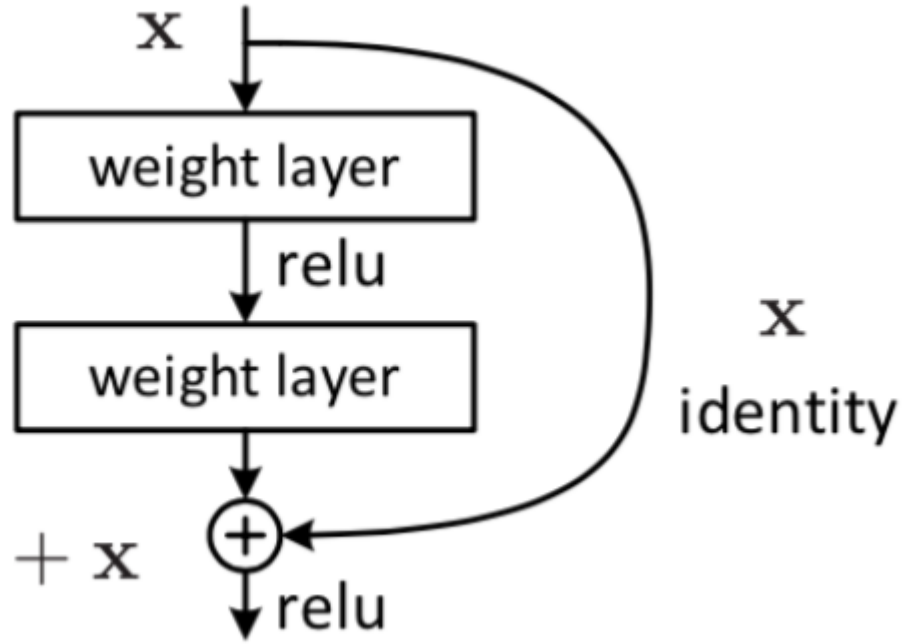
- Identity Mapping을 하는 Layer를 쌓으면 아무일도 하지 않는 Layer가 추가되는 것이기 때문에 적어도 성능이 더 떨어지지는 않을 것이라는 생각

Underlying Mapping을 직접 학습하지 않고 Residual Mapping에 fit하여 학습하자!

Residual Mapping : 최적의 H(x)를 찾는 것이 목적

H(x) - x <= $\mathcal{F}(\mathbf{x})$

H(x) => $\mathcal{F}(\mathbf{x}) + \mathbf{x}$



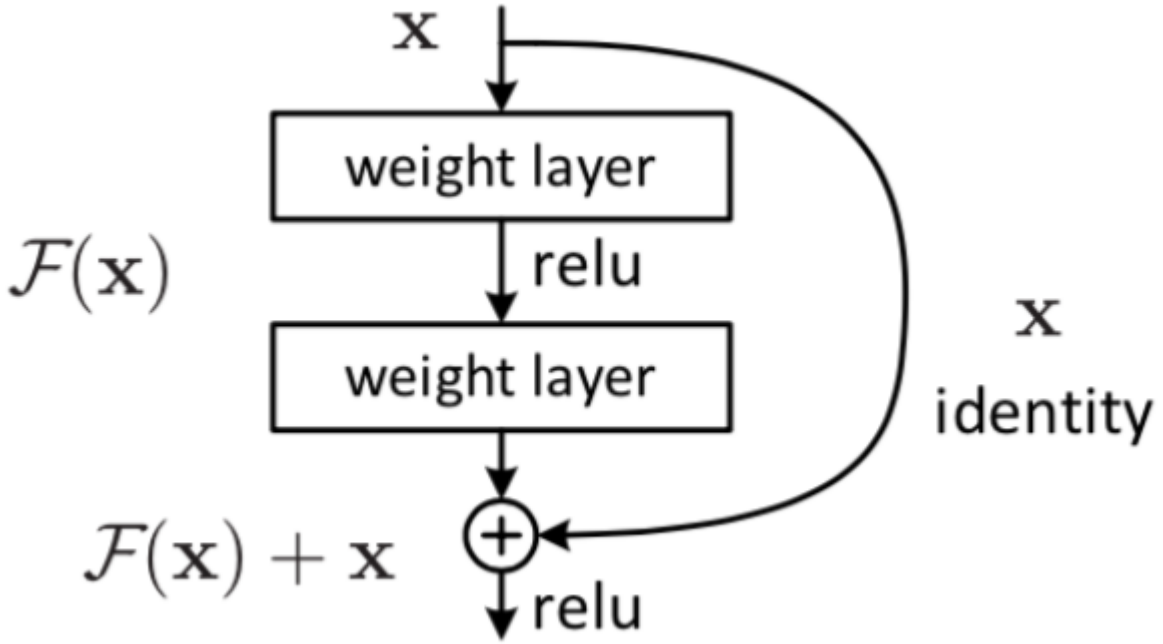
Residual Mapping을 Optimize하는 것이 더 쉽다!

H(x)에 대한 optimal solution이 identity mapping이라고 가정하면,
H(x)를 x가 되도록 학습하는 것보다는 F(x) = H(x) – x = 0이 되도록 학습하는 것이 더 쉬울 것이다.

즉, residual (입력인 x와의 잔차) F(x)가 0이 되도록 하는 것을 목적으로 한다.

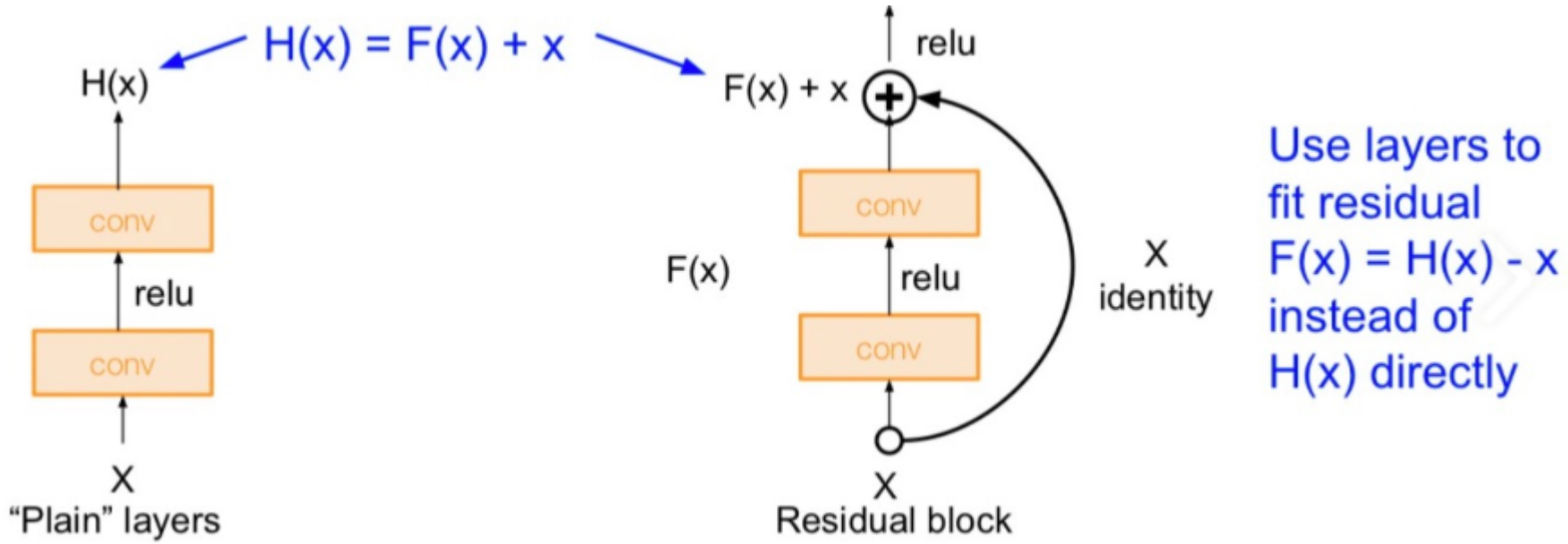
Residual Block : Skip Connection (Shortcut Connection)으로 구현

-> 한 개 이상의 layer를 skipping하는 것



Identity Mapping에 의해 Block의 Input(x)이 Output에 그대로 더해지게 된다.

Residual Block : Skip Connection (Shortcut Connection)으로 구현



Plain layers: original mapping

Residual block: identity mapping

출처: CS231n

Residual Learning의 장점

1. Vanishing Gradient 문제 해결!
- 단순 더하기로 이루어져있기 때문에 Backpropagation의 gradient가 1이 되고,
  이는 Output에서 Input에 가까운 Layer까지의 Loss가 0이 안되게 한다.

2. 계산 비용 x!
   추가 Parameter x!
- 단순 더하기이므로

3. Residual Network의 경우 깊어져도 성능을 높일 수 있어 degradation 문제 해결

Contents

Residual Learning

앞에서도 이야기했듯이,
깊은 층을 쌓아서 생기는 성능 저하 문제를 해결하기 위해서
stacked layer를 F(x) := H(x) – x에 mapping하여 original mapping을 F(x) + x로 설정

H(x)가 여러 비선형 layer로 이루어져 복잡한 함수에 근사한다고 가정할 때,
여기에서 H(x)를 학습하는 대신 H(x) – x = F(x)를 학습하는 것이 더 쉬울 것이라고 가정

Identity mapping by Shortcuts

stacked layers마다 residual learning을 사용

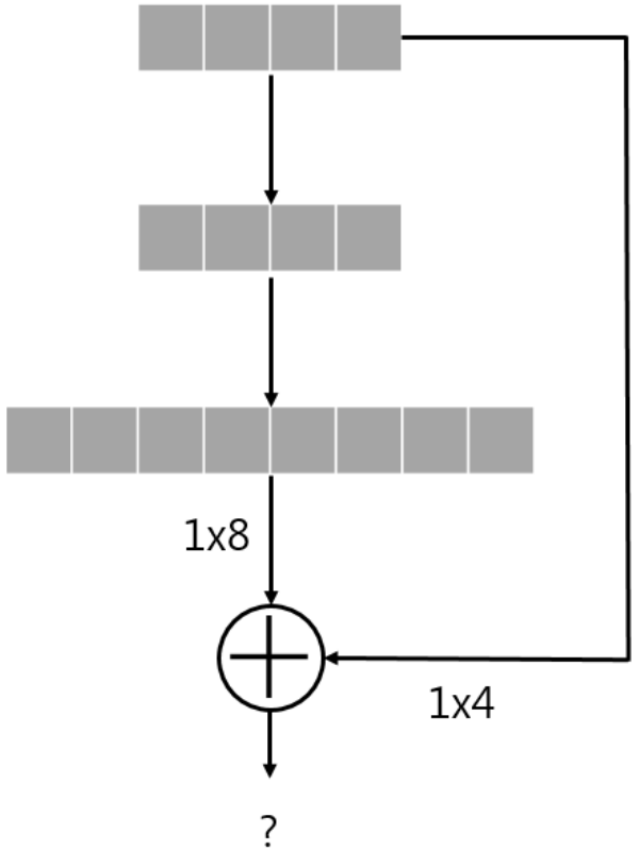
$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$ (x:input, y:output, F: residual mapping)

위에서 F + x 연산에서 둘의 dimension이 같아야 하며, 이를 위해 linear projection Ws 수행

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

첫번째 식에서 Ws를 사용할 수 있으나,
identity mapping만으로 충분하다는 것을 실험을 통해 보여준다.
따라서 Ws는 dimension matching의 용도로만 사용한다.

Identity mapping by Shortcuts



왼쪽의 경우,

shortcut을 통한 output은 1*4

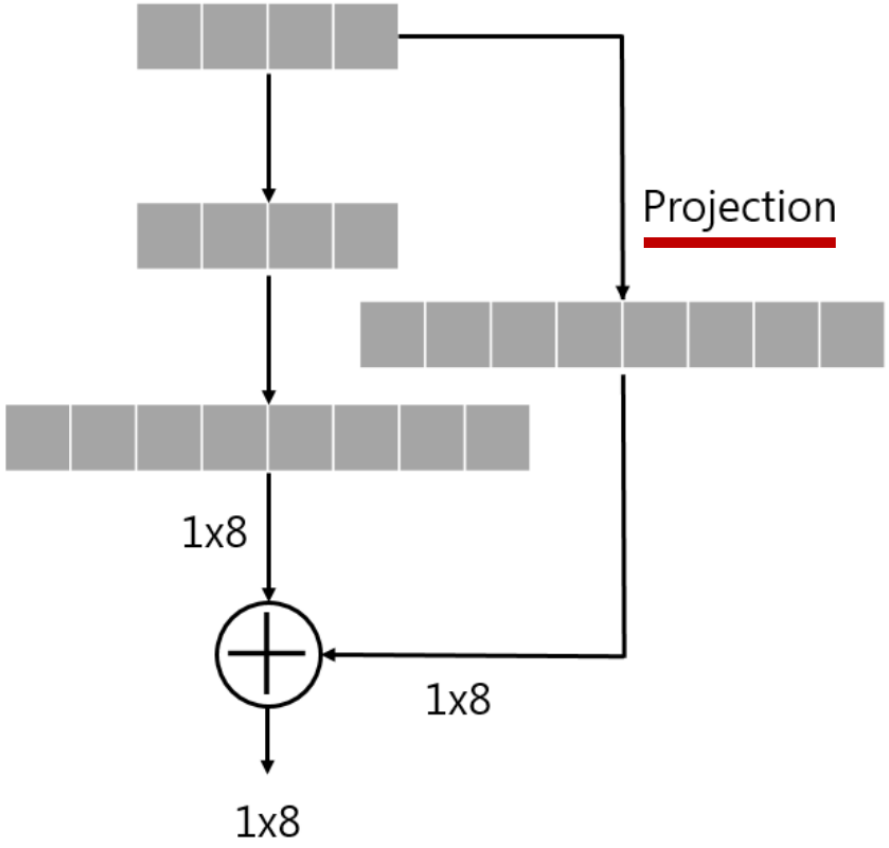
2-layer을 통한 output은 1*8로

dimension이 다르기 때문에 덧셈을 할 수 없다.

Identity mapping by Shortcuts

따라서 다른 dimension을 맞추기 위해,

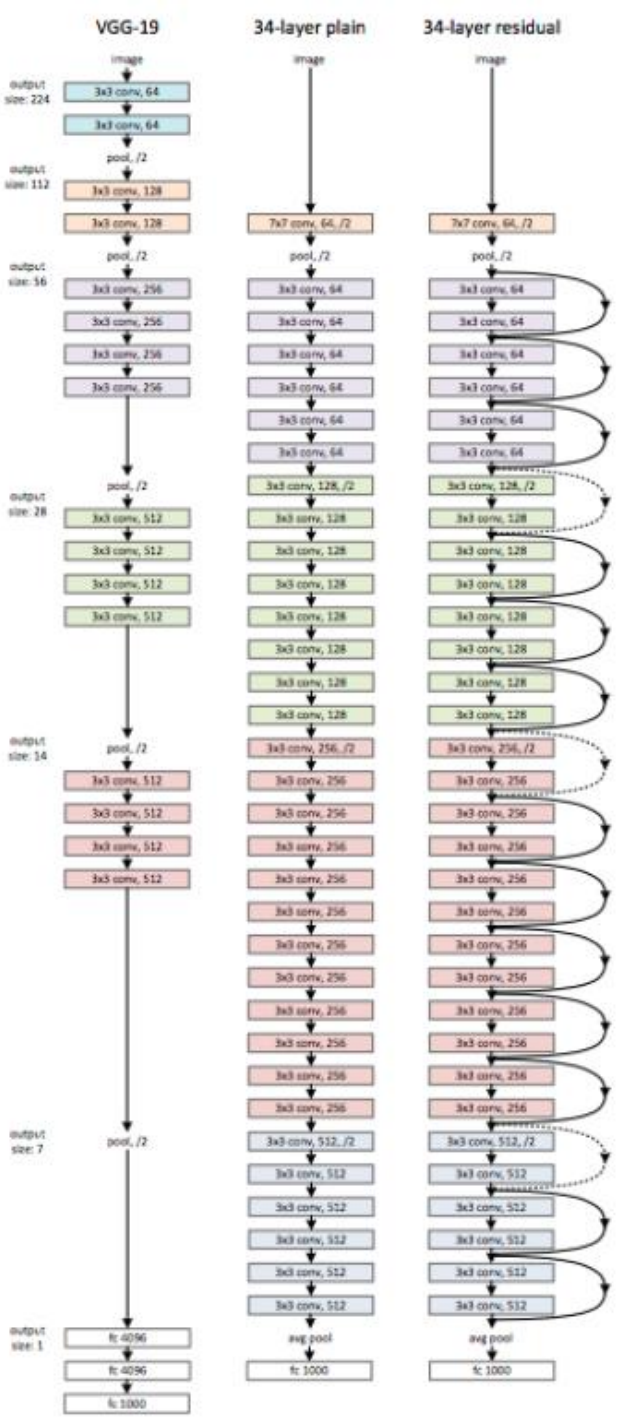
projection의 과정을 거쳐 덧셈을 진행한다.



Network Architectures

ImageNet의 data를 이용해

Plain network와 residual network 비교



Unit 03 | Deep Residual Learning

1. Plain Network

VGGNet의 영향을 많이 받음
conv layer는 3*3의 filter를 가지며 두 가지의 규칙에 의해 design

1) 동일한 output feature map size에 대해, layer는 동일한 filter 수를 가진다.

2) feature map size가 절반인 경우,
layer당 시간복잡도를 보존하기 위해 filter의 수를 2배로 한다.

stride=2인 conv layer를 통해 downsampling
network의 마지막에는 1000-way FC layer로 구성

2. Residual Network

plain network를 기반으로 short connection 삽입

identity shortcut은
- input과 output이 동일한 dimension인 경우에는, 바로 사용된다.

- input과 output이 동일하지 않을 때에는 (증가할 때에는),
  1) zero padding을 추가하여 dimension을 맞춘 후 identity mapping 수행한다
  2) projection을 통해 dimension을 맞춰준다

Implementation

- Data Augmentation으로 standard color augmentation 사용
- 각각의 conv layer와 activation 사이에 batch normalization 사용하며
  he initialization으로 weight 초기화
- batch normalization에 근거하여 dropout 사용 x
- SGD를 사용하며 learning rate는 0.1에서 시작하여
  local minimum을 만나거나 loss가 진동하는 경우 1/10씩 감소
- weight decay = 0.0001, momentum = 0.9
- mini-batch size = 256, iteration = 60 * 10^4

Contents

ImageNet Classification

1000개의 class를 가진 ImageNet 2012 classification dataset 사용

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

ImageNet Classification

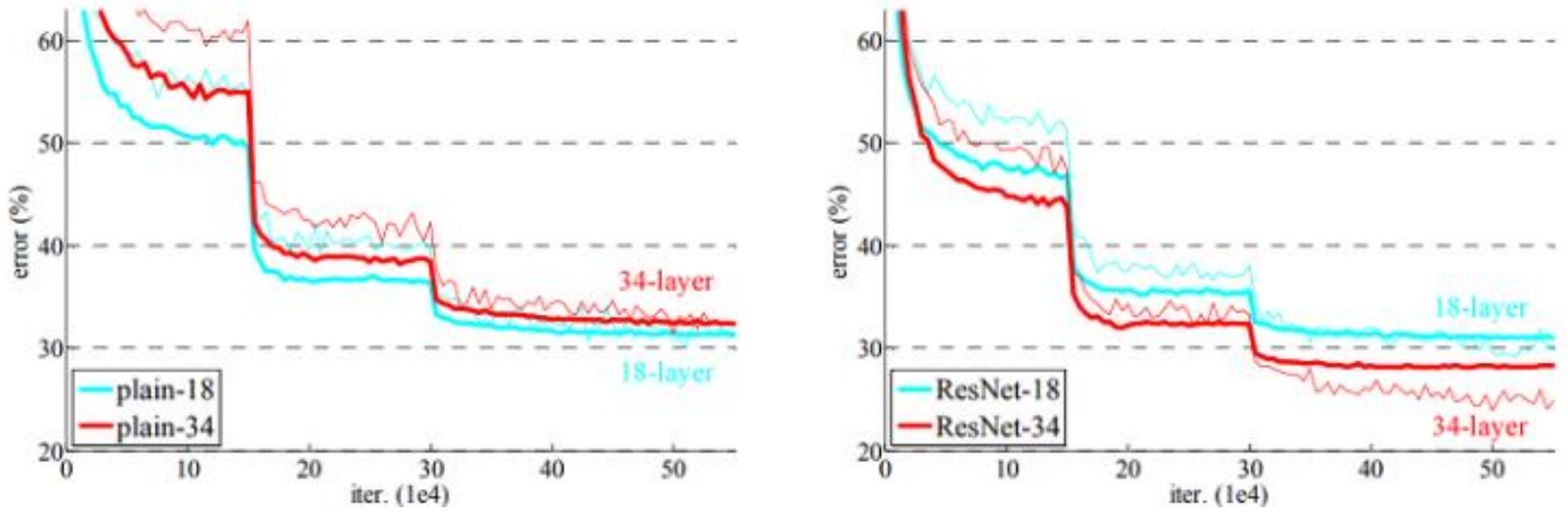
	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

18 layer, 34 layer의 plain network에서 비교해보면,

34 layer의 validation error가 더 높게 나오는 degradation 문제 발견

(이는 vanishing gradient의 문제는 아니며, 낮은 수렴률에 의한 이유로 보았다.)

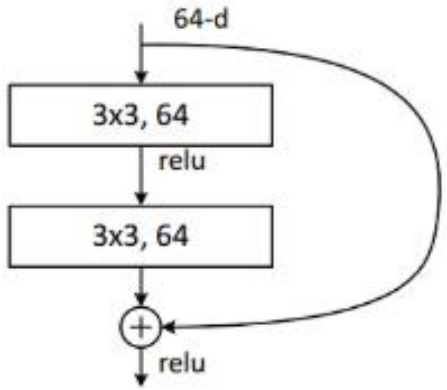
ImageNet Classification

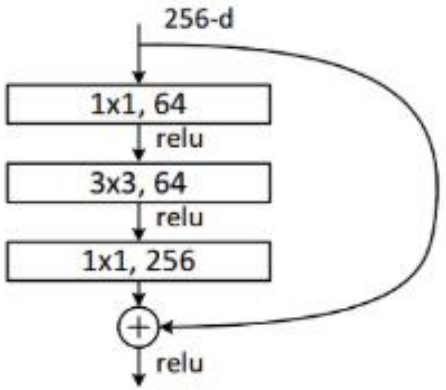


반면에 residual network의 경우 layer가 깊어질수록 error rate가 더 줄어들었다!

또 optimization 또한 더욱 빠르게 수렴하는 것을 알 수 있다!

ImageNet Classification





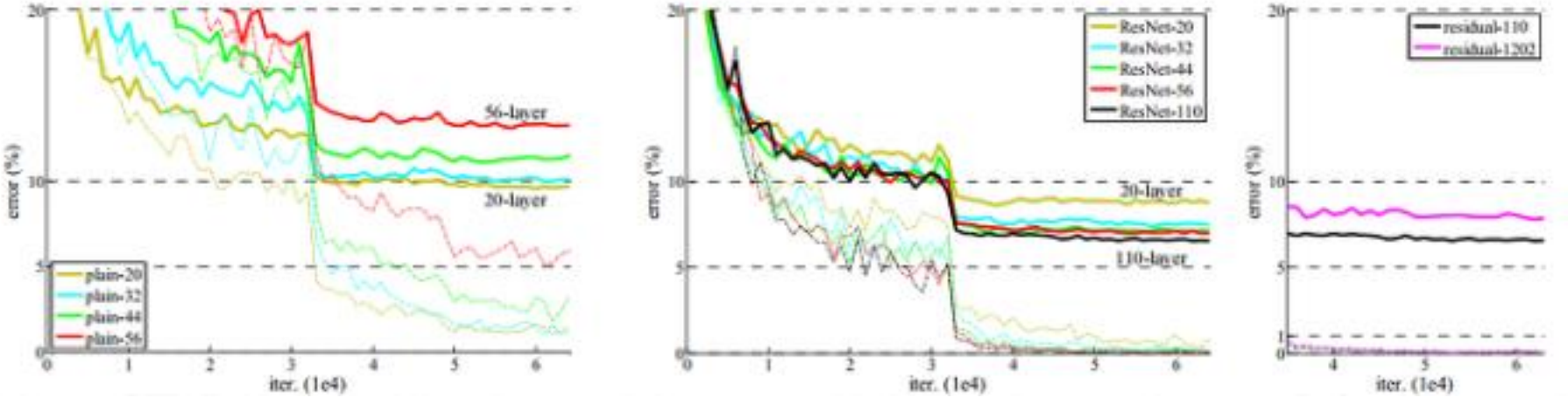
더 깊은 모델을 위해 2 layer가 아닌 3 layer 사용

1*1, 3*3, 1*1의 conv로 이루어져 있다.

첫번째 1*1은 dimension을 줄이고, 마지막 1*1은 dimension을 늘리기 위해 쌓았으며,

중간의 3*3 conv는 dimension이 줄어들었다가 다시 증가하는 bottleneck 구조

CIFAR-10

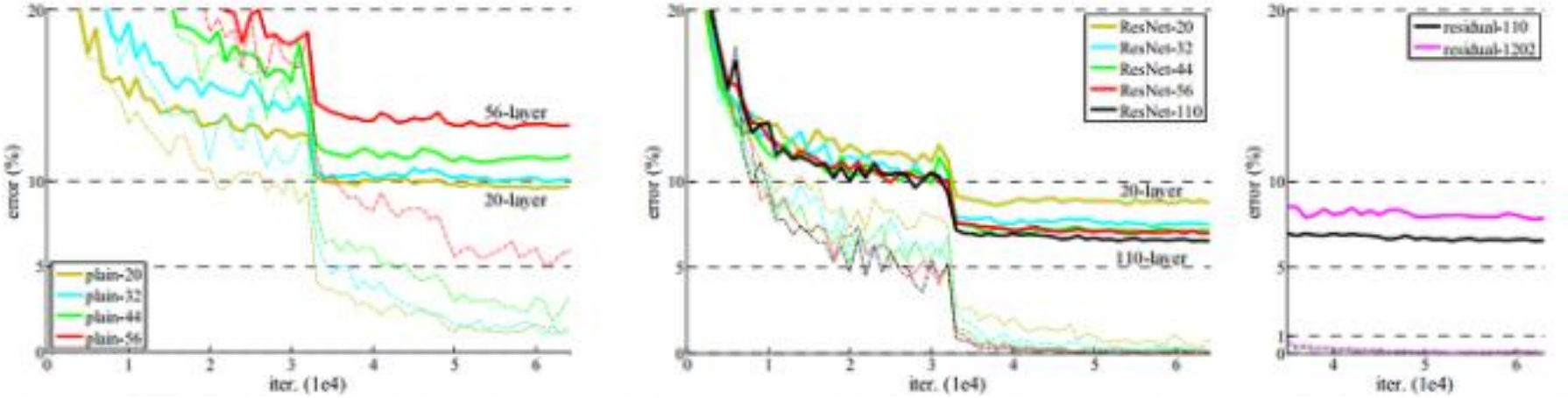


CIFAR-10 데이터셋에 대한 학습 결과

plain의 경우 20 > 32 > 44 > 56의 순서로 성능이 좋았으나,

residual의 경우 110 > 56 > 44 > 32 > 20으로 깊을수록 좋은 성능을 보였다.

CIFAR-10



반면 1202 layer ResNet의 경우 110 layer ResNet에 비해 성능이 많이 떨어졌다.

이는 모델이 불필요하게 커서 나타나는 overfitting 현상으로 주장한다.

Object Detection on PASCAL and MS COCO

기존의 VGG를 사용하던 Faster R-CNN에 ResNet을 이용하여 Object Detection을 사용하였다.

그 결과, 성능이 크게 향상되었다.

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.

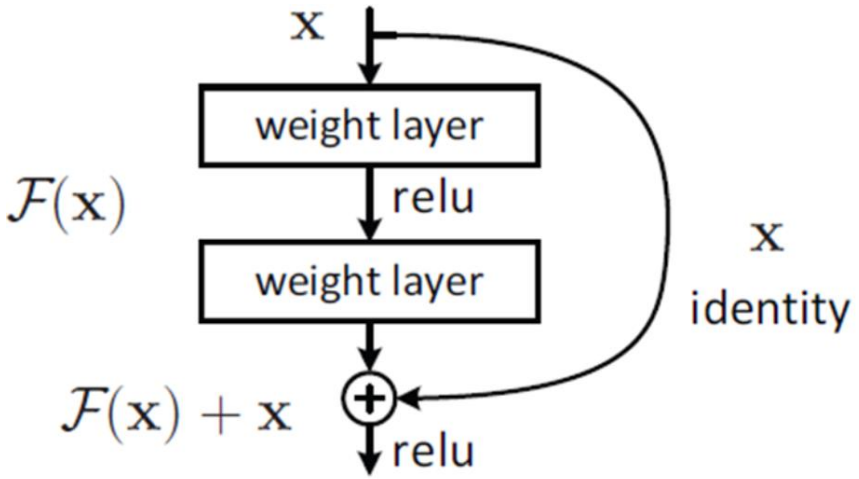
마무리

ResNet 한 장 정리

깊은 망에서 성능 저하 현상이 나타나는 것을 해결하기 위해
Residual Learning을 진행하였다!
이는 F(x) = H(x) − x = 0이 되는 방향으로 학습하는 것이다.

input이 그대로 output에 연결되므로 parameter 수에 영향이 없으며,
덧셈이 늘어나는 것 외에 연산량 증가 또한 없다.

이를 통해 깊은 망도 쉽게 optimize 가능하며,
늘어난 깊이로 인해 정확도를 높이는 효과 또한 가져왔다.



Reference

https://sike6054.github.io/blog/paper/first-post/#1-introduction

https://curaai00.tistory.com/1

https://leechamin.tistory.com/184

CS231n

들어주셔서 감사합니다.