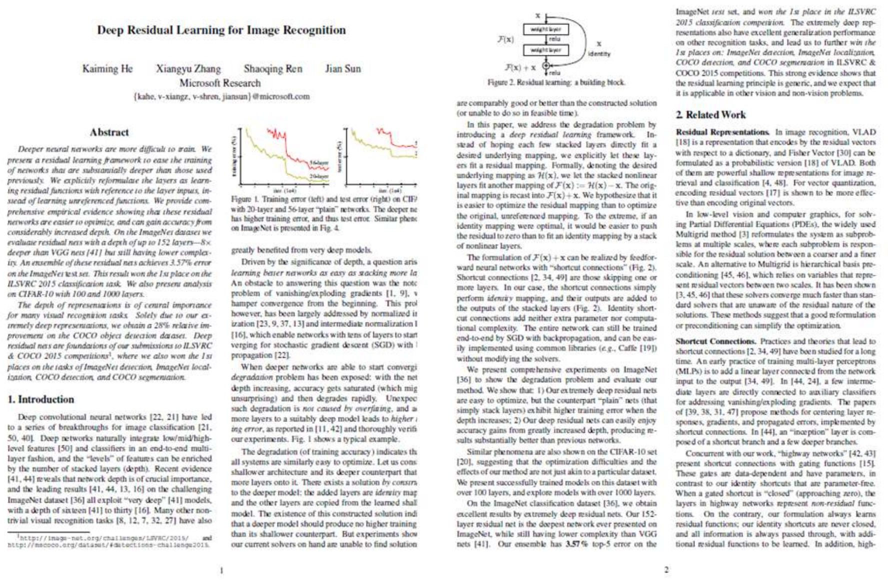
Residual Neural Network (ResNet)



윤예중

dbsdb2222@g.skku.edu

Computer Vision

2023/03/07

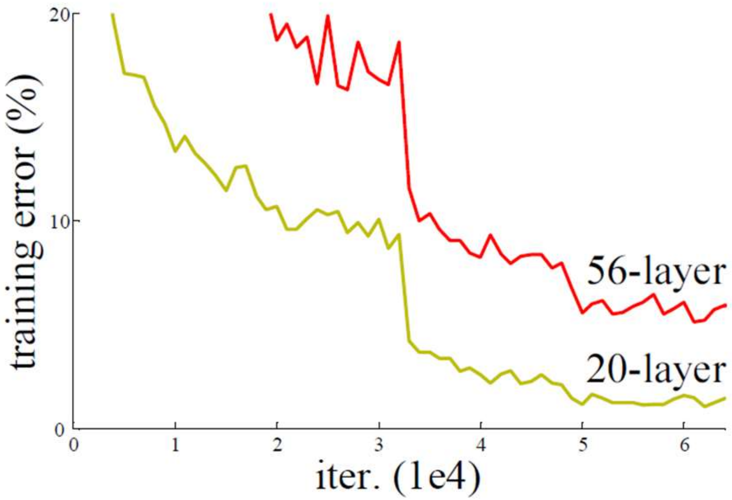
TRAIN AND TEST

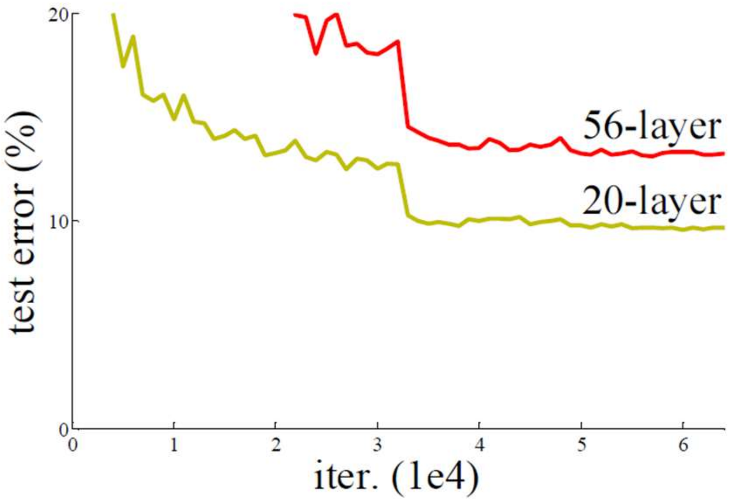
Contents

TRAIN AND TEST

Degradation Problem

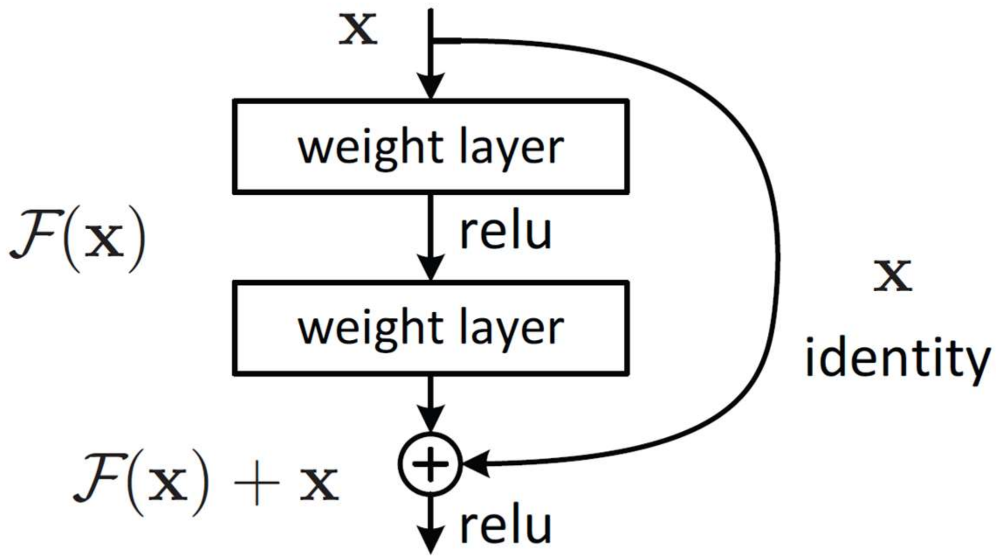
Plain Network





- CIFAR-10 Dataset
- 20-layer VS 56-layer
- Error : 56-layer > 20-layer (both training and test)
- Not overfitting
- Gradients vanishing/exploding

Shortcut Connection/Identity Mapping



- Identity : 입력값을 그대로 전달함
- $F(x) = H(x) - x$
- H$(x) = F(x) + x$ (shortcut connection)
- 하나 이상의 layer를 skip
- $F(x)$: Model, x : identity(자기 자신)
- ReLU : function

TRAIN AND TEST

Residual Learning

Equations

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

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$ Eq. (2)

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$ Eq. (3)

Eq. (2)

- σ : ReLU, W_1, W_2 : Weights

Eq. (1)

- x : input, y : output
- $F(x, \{W_i\}) + x$: 학습될 residual mapping
- x와 F의 차원이 같아야 함

Eq. (3)

- W_s : linear projection(차원을 맞추기 위함)
- Eq.(1)으로 충분

TRAIN AND TEST

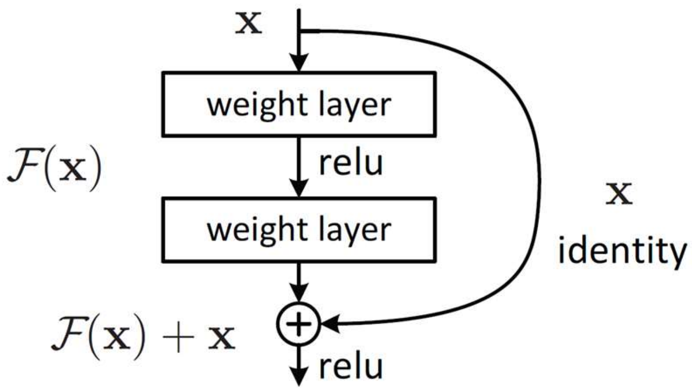
Residual Learning

Single/Multiple Layer

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

If F is single layer

$$\mathbf{y} = W_1\mathbf{x} + \mathbf{x}$$

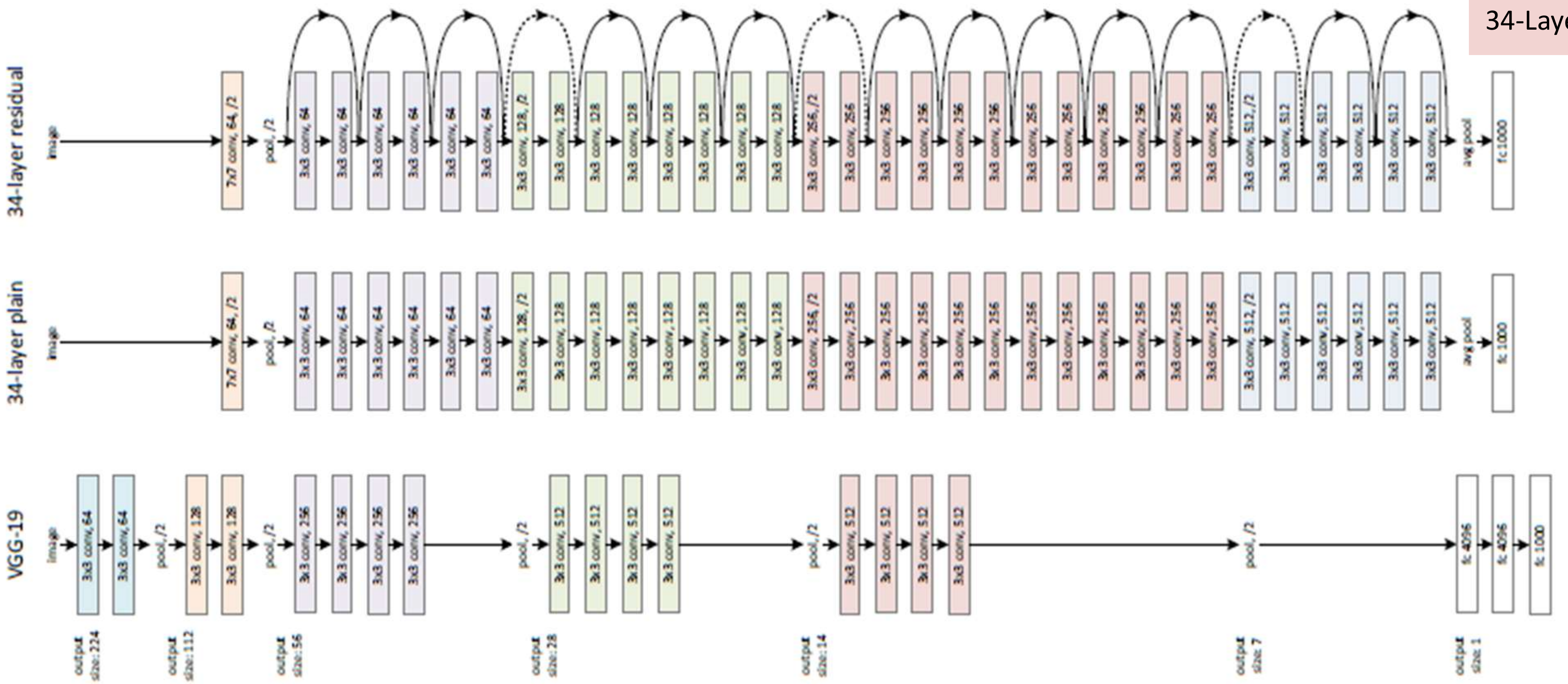


- Single Layer : Linear Equation이 되므로 shortcut의 장점을 살릴 수 없다
- $F(x, \{W_i\})$를 이용하여 multiple convolution layer로 활용 가능

Network Architectures

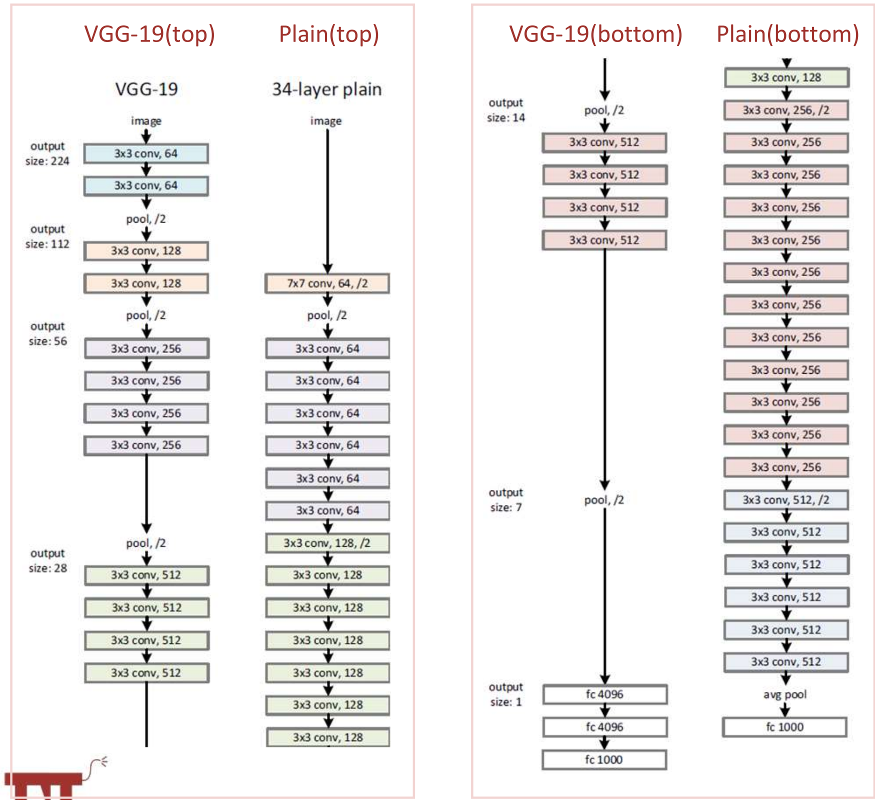
VGG vs Plain vs ResNet

Model	Performance (FLOPs)
VGG-19	19.6 billion FLOPs
34-Layer Plain	3.6 billion FLOPs
34-Layer Residual	3.6 billion FLOPs



Network Architectures

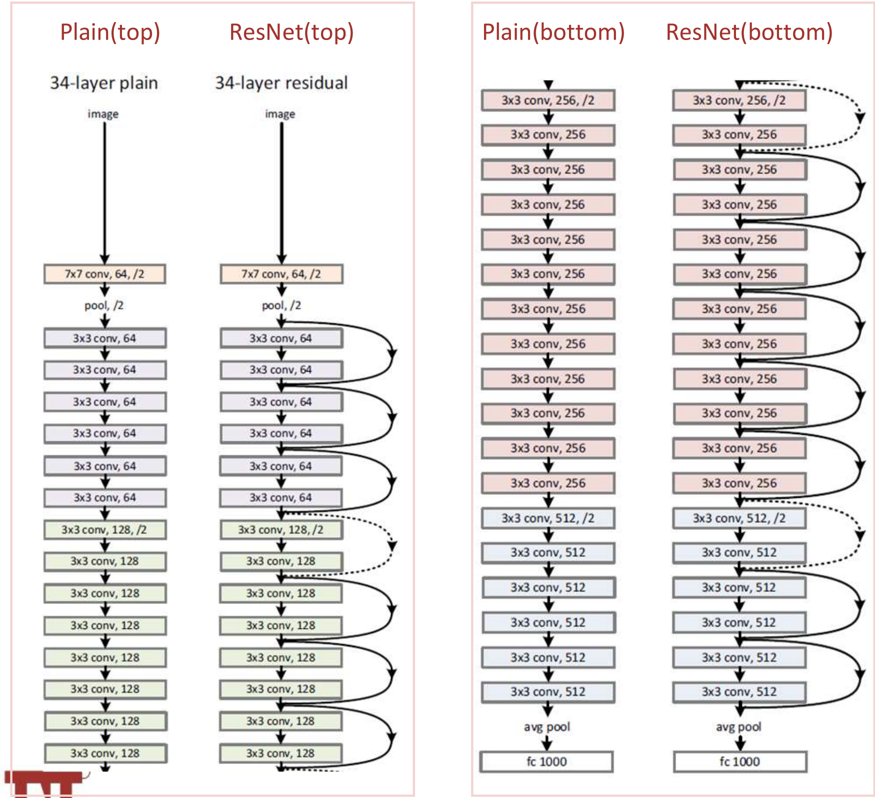
Plain Network



- Plain network (34-layer) : VGG-net 기반으로 만들어짐
- Convolution Layer : 동일한 feature map 사이즈와 동일한 filter 수 사용 -> 3x3 filters
- Output Layer : global average pooling + 1000-way fully-connected(with softmax)

Network Architectures

Residual Network



- Plain network 기반 + shortcut connection
- Input, output dimension이 같아야 함

  (1). Zero padding (no extra parameter)

  (2). Linear projection

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

Implementation

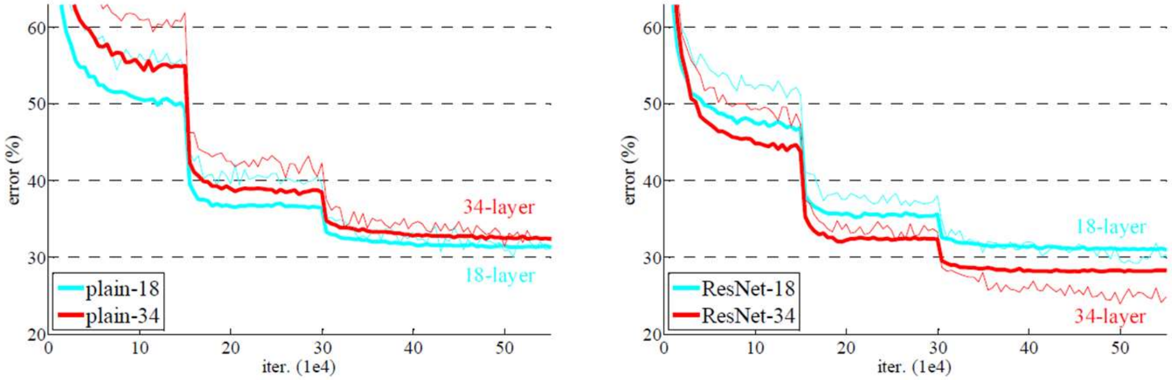
Subtitle

- 224x224 crop (randomly sampled)
- Batch Normalization (after convolution before activation
- Initialize Weights
- Stochastic Gradient Descent (SGD, mini-batch size of 256)
- Learning Rate : 0.1으로 시작, 에러 발생 시 1/10배씩
- Iteration : 60×10^4
- Weight Decay : 0.0001, Momentum : 0.9, no dropout

TRAIN AND TEST

Experiments

ImageNet Classification



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

- Plain error : 18-layers < 34-layers
- ResNet error : 18-layers > 34-errors
- 34-layer에서의 error : Plain > ResNet
- 18-layer에서의 error는 비슷, ResNet이 더 빨리 수렴

TRAIN AND TEST

Experiments

Identity vs Projection Shortcuts

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

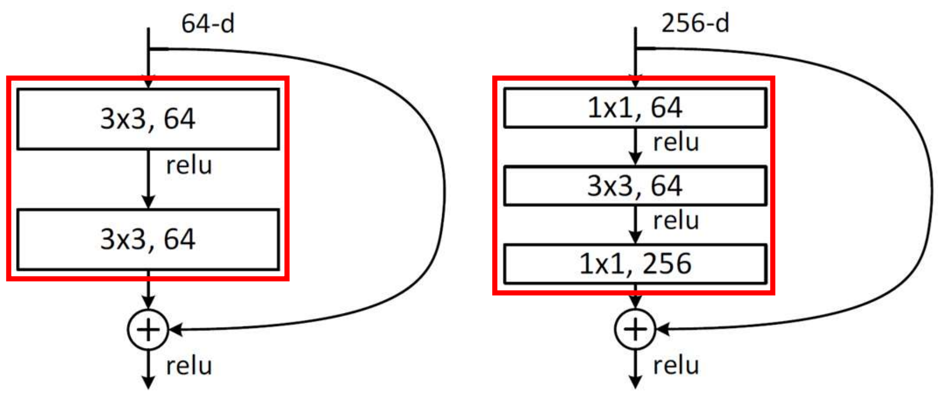
Case	특징
A	차원 증가를 위해 zero-padding 사용 (no extra parameter)
B	차원 증가가 필요할 때는 projection shortcut, 나머지는 identity shortcut
C	모든 shortcut에 projection shortcut 사용

- Error : A > B > C
- A > B : A에 residual learning이 이뤄지지 않기 때문
- B > C : C에서 extra parameter를 사용했기 때문
- -> C가 성능은 가장 좋지만 time/memory complexity 문제로 사용X

TRAIN AND TEST

Experiments

Bottleneck Architectures



General building block   Bottleneck building block

- Deeper Network에서는 학습시간이 느려지기 때문에 Bottleneck 구조 사용
- F function에 3개의 layer 사용
- 1x1 -> 3x3 -> 1x1 형식의 layer 사용
- 2-layer의 input/output dimension : 3x3
- 3-layer의 input/output dimension : 1x1

TRAIN AND TEST

Experiments

Bottleneck Architectures

Model		Performance (FLOPs)
ResNet	18-layers	1.8 billion
	34-layers	3.6 billion
	50-layers	**3.8 billion**
	101-layers	**7.6 billion**
	152-layers	**11.3 billion**
VGG-net	16-layers	15.3 billion
	19-layers	19.6 billion

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**

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**

- Layer가 깊어질수록 기존 model에 bottleneck 더 많이 추가
- 성능 : VGG-net보다 좋음
- Error : 층이 깊어질수록 error가 줄어듦

TRAIN AND TEST

Experiments

Etc.

method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Classification on CIFAR-10

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**

Object Detection on PASCAL VOC

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	**48.4**	**27.2**

Object Detection on COCO

TRAIN AND TEST

TRAIN AND TEST