

# Residual Neural Network (ResNet)

## Deep Residual Learning for Image Recognition

Kaiming He Xiangyong Zhang Shaoqing Ren Jian Sun  
Microsoft Research  
(khe, v-tang, v-shen, jiansun)@microsoft.com

### Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—3x deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.5% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 200 layers.

The depth of representations is of central importance for many visual recognition tasks. Soleyly due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions<sup>1</sup>, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

### 1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 50, 49]. Deep networks naturally integrate low/mid-high-level features [50] and classifiers in an end-to-end multi-layer fashion, and the “level” of features can be enriched by the number of stacked layers (depth). Recent evidence [41, 49] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 10] on the challenging ImageNet dataset [16] all exploit “very deep” [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-visual recognition tasks [8, 12, 7, 32, 27] have also

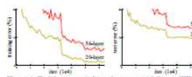


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20 layer and 152 layer “plain” networks. The deeper net has higher training error, and thus test error. Similar plots on ImageNet is presented in Fig. 4.

greatly benefited from very deep models. Driven by the significance of depth, a question arises: learning better networks as easy as stacking more layers. An obstacle to answering this question was the notorious problem of “vanishing/exploding gradients” [1, 9], a hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 11] and intermediate normalization [116], which enable networks with tens of layers to start learning for stochastic gradient descent (SGD) with 1 propagation [22].

When deeper networks are able to start converge, degradation problem has been exposed: with the net depth increasing, accuracy gets saturated (which may be surprising), and then degrades rapidly. Unchecked such degradation is not caused by overfitting, and as more layers to a suitably deep model leads to higher test error, as reported in [11, 42] and thoroughly verified our experiments. Fig. 1 shows a typical example.

The degradation (of training accuracy) indicates that all systems are similarly easy to optimize. Let us consider shallower architectures and its deeper counterpart that more layers onto it. There exists a solution by contrast, to the deeper model: the added layers are identity maps and the other layers are copied from the learned shallower model. The existence of this constructed solution (indicated that a deeper model should produce no higher training than its shallower counterpart. But experiments show our current solvers on hand are unable to find solution

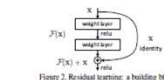


Figure 2. Residual learning: a building block. are comparably good or better than the constructed solution (or unable to do so in feasible time).

In this paper, we address the degradation problem by introducing a deep residual learning framework. Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Formally, knowing the desired underlying mapping as  $\mathcal{H}(x)$ , we let the stacked nonlinear layers fit another mapping of  $\mathcal{F}(x) := \mathcal{H}(x) - x$ . The original mapping is recast into  $\mathcal{F}(x) + x$ . We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.

The formulation of  $\mathcal{F}(x) + x$  can be realized by feedforward neural networks with “shortcut connections” (Fig. 2). Shortcut connections [2, 34, 49] are those skipping one or more layers. In our case, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers (Fig. 2). Identity shortcut connections add neither extra parameter nor computational complexity. The entire network can still be trained end-to-end by SGD with backpropagation, and can be easily implemented using common libraries (e.g., Caffe [19]) without modifying the solvers.

We present comprehensive experiments on ImageNet [36] to show the degradation problem and evaluate our method. We show that: 1) Our extremely deep residual nets are easy to optimize, but the counterpart “plain” nets (that simply stack layers) exhibit higher training error when the depth increases; 2) Our deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks.

Similar phenomena are also shown on the CIFAR-100 set [20], suggesting that the optimization difficulties and the effects of our method are not just akin to a particular dataset. We present successfully trained models on this dataset with over 100 layers, and explore models with over 1000 layers.

On the ImageNet classification dataset [36], we obtain excellent results by extremely deep residual nets. Our 152-layer residual net is the deepest network ever presented on ImageNet, while still having lower complexity than VGG nets [41]. Our ensemble has 3.5% top-5 error on the

ImageNet test set, and won the 1st place in the ILSVRC 2015 classification competition. The extremely deep representations also have excellent generalization performance on other recognition tasks, and lead us to further win the 1st places on: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in ILSVRC & COCO 2015 competitions. This strong evidence shows that the residual learning principle is generic, and we expect that it is applicable in other vision and non-vision problems.

### 2. Related Work

**Residual Representations.** In image recognition, VLAD [18] is a representation that encodes by the residual vectors with respect to a dictionary, and Fisher Vector [30] can be formulated as a probabilistic version [18] of VLAD. Both of them are powerful shallow representations for image retrieval and classification [4, 46]. For vector quantization, encoding residual vectors [17] is shown to be more effective than encoding original vectors.

In low-level vision and computer graphics, for solving Partial Differential Equations (PDEs), the widely used Multigrid method [3] reformulates the system as subproblems at multiple scales, where each subproblem is responsible for the residual solution between a coarser and a finer scale. An alternative to Multigrid is hierarchical basis preconditioning [45, 46], which relies on variables that represent residual vectors between two scales. It has been shown [3, 45, 46] that these solvers converge much faster than standard solvers that are unaware of the residual nature of the solutions. These methods suggest that a good reformulation or preconditioning can simplify the optimization.

**Shortcut Connections.** Practices and theories that lead to shortcut connections [2, 34, 49] have been studied for a long time. An early practice of training multi-layer perceptrons (MLPs) is to add a linear layer connected from the network input to the output [34, 49]. In [44, 24], a few intermediate layers are directly connected to auxiliary classifiers for addressing vanishing/exploding gradients. The papers of [39, 38, 31, 47] propose methods for centering layer responses, gradients, and propagated errors, implemented by shortcut connections. In [44], an “shortcut” layer is composed of a shortcut branch and a few deeper branches.

Concurrent with our work, “highway networks” [42, 43] present shortcut connections with gating functions [15]. These gates are data-dependent and have parameters. In contrast to our identity shortcuts that are parameter-free. When a gated shortcut is “closed” (approaching zero), the layers in highway networks represent non-residual functions. On the contrary, our shortcuts always learn residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned. In addition, high-

윤예중

dbsdb2222@g.skku.edu

Computer Vision

2023/03/07



TRAIN AND TEST

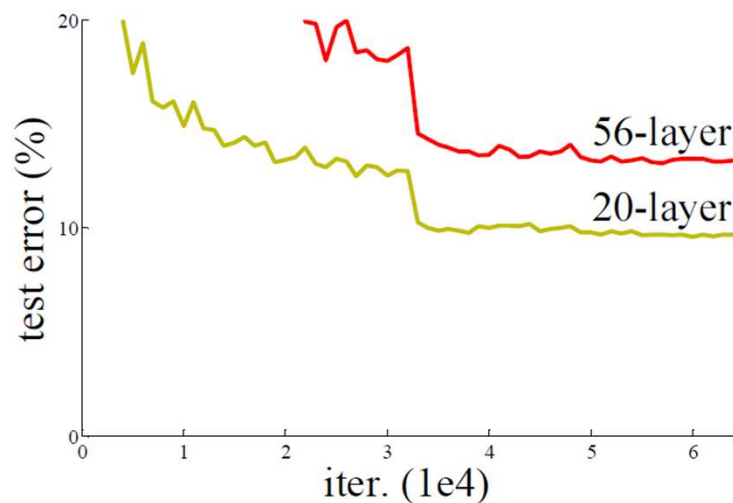
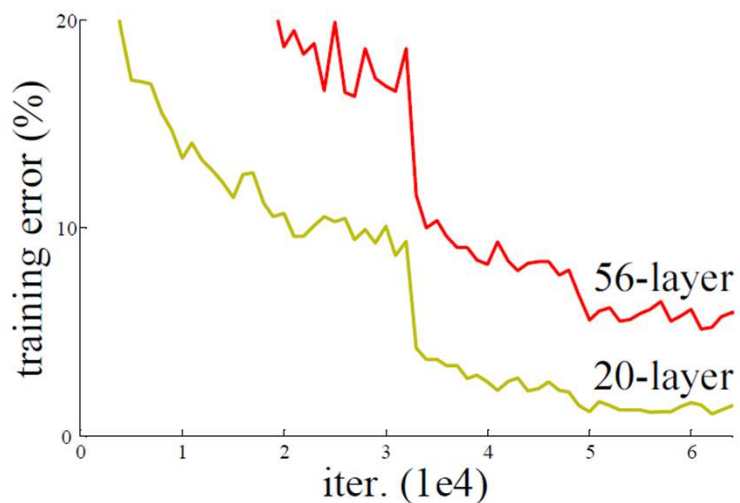
# Contents

---

- Degradation Problem
- Shortcut Connection/Identity Mapping
- Residual Learning
- Network Architectures
- Implementation
- Experiments

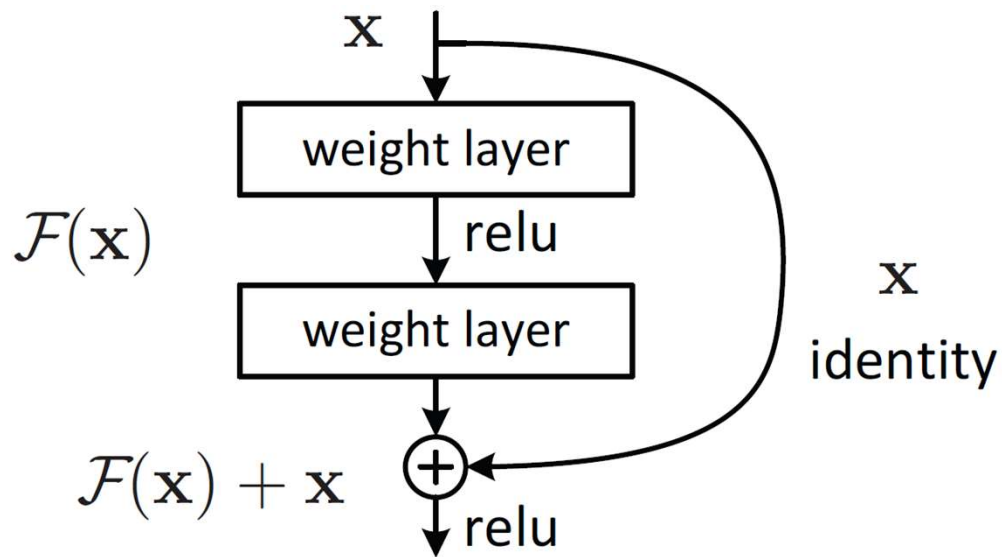
# Degradation Problem

## 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

# Residual Learning

## Equations

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

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

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

Eq. (2)

- $\sigma$  : ReLU,  $W_1, W_2$  : Weights

Eq. (1)

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

Eq. (3)

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

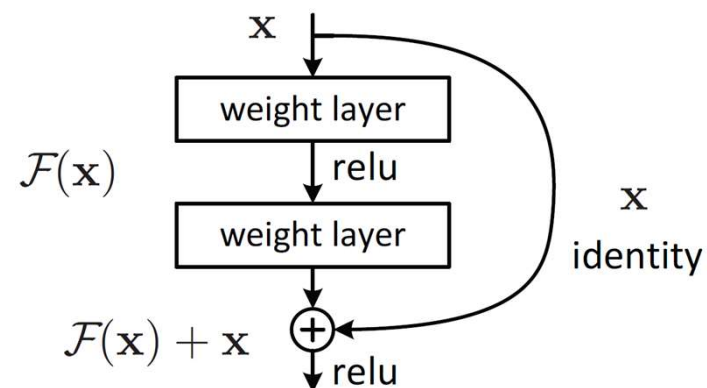
# Residual Learning

## Single/Multiple Layer

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

If  $F$  is single layer

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

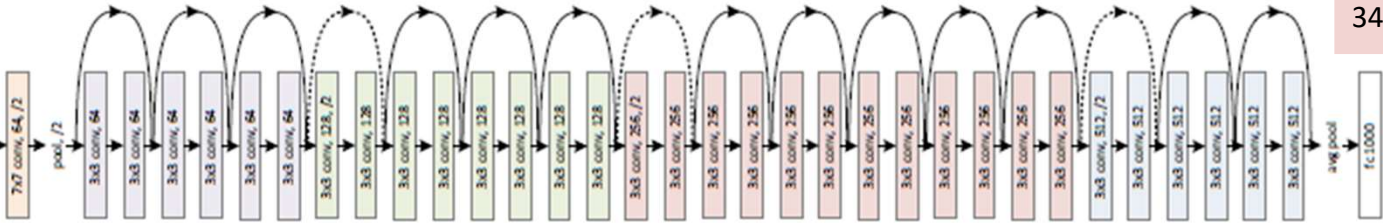


- Single Layer : Linear Equation이 되므로 shortcut의 장점을 살릴 수 없다
- $F(x, \{W_i\})$ 를 이용하여 multiple convolution layer로 활용 가능

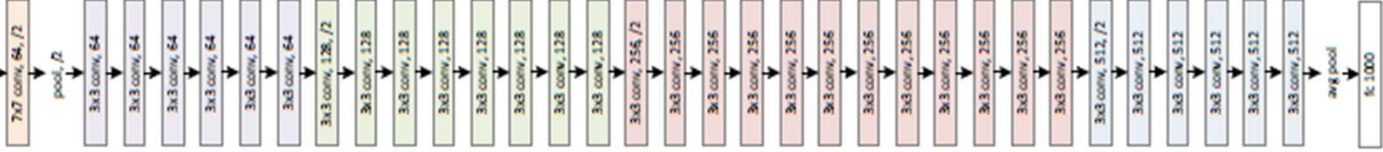
# Network Architectures

## VGG vs Plain vs ResNet

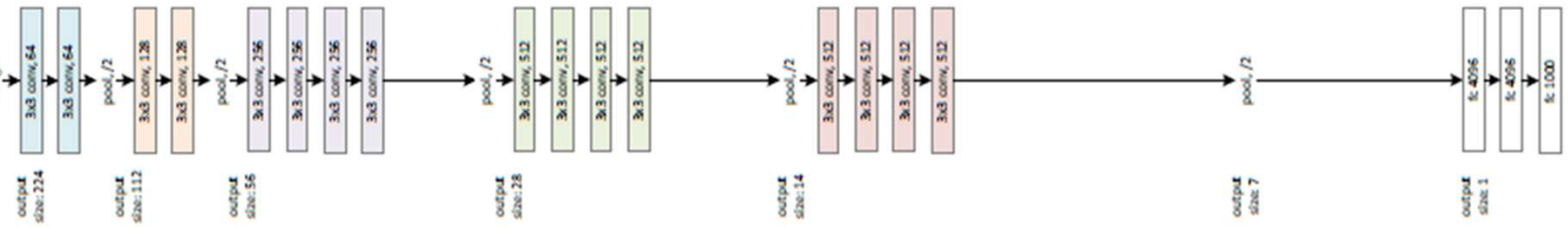
34-layer residual



34-layer plain



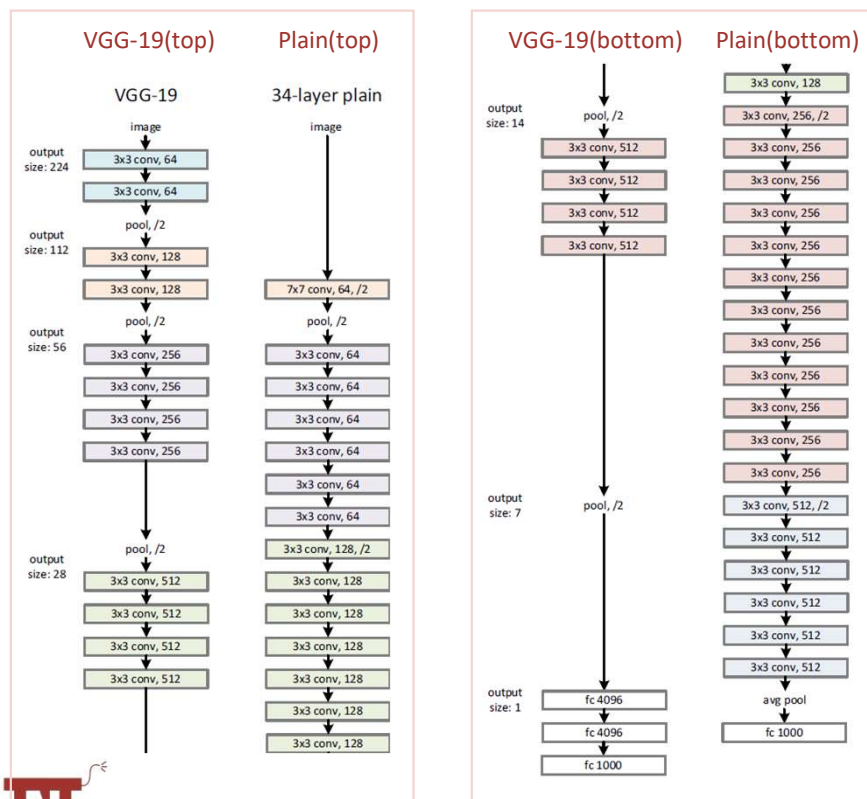
VGG-19



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

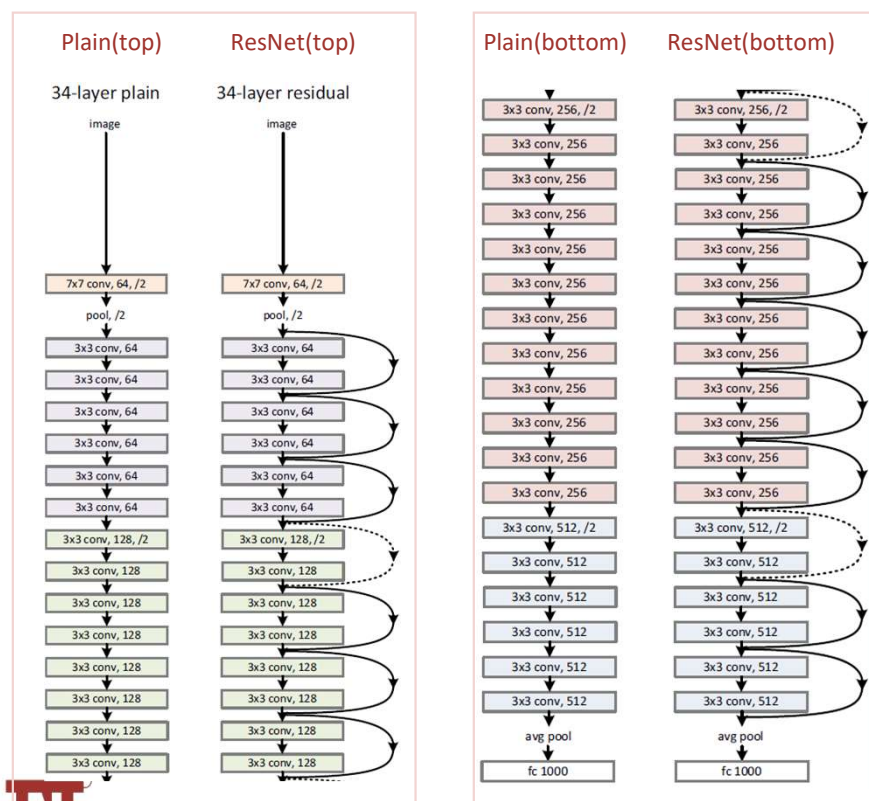


- Plain network (34-layer) : VGG-net 기반으로 만들어짐
- Convolution Layer : 동일한 feature map 사이즈와 동일한 filter 수 사용 -> 3x3 filters
- Output Layer : global average pooling + 1000-way fully-connected(with softmax)



# Network Architectures

## Residual Network



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

(1). Zero padding (no extra parameter)

(2). Linear projection

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

# Implementation

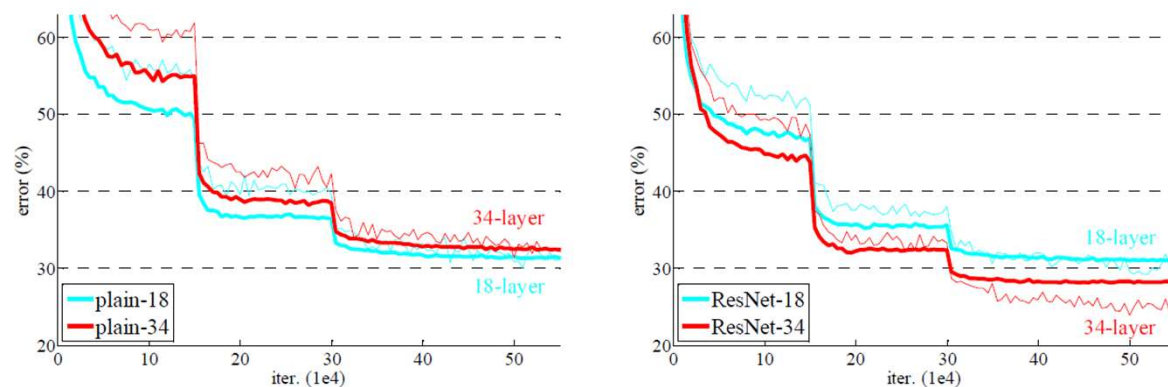
---

## 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 :  $60 \times 10^4$
- Weight Decay : 0.0001, Momentum : 0.9, no dropout

# Experiments

## ImageNet Classification



	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	<b>25.03</b>

- 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

# Experiments

## Identity vs Projection Shortcuts

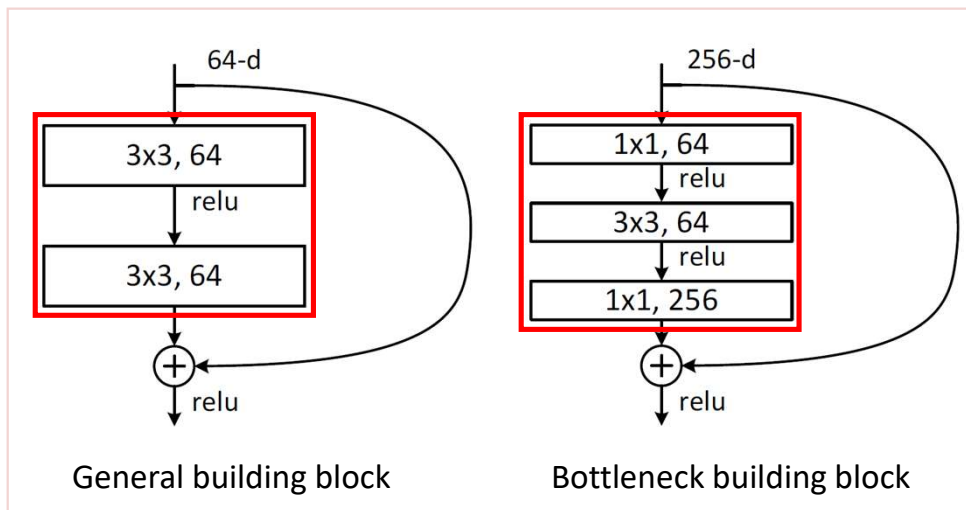
Case	특징
A	차원 증가를 위해 zero-padding 사용 (no extra parameter)
B	차원 증가가 필요할 때는 projection shortcut, 나머지는 identity shortcut
C	모든 shortcut에 projection shortcut 사용

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

- 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

# Experiments

## Bottleneck Architectures



- Deeper Network에서는 학습시간이 느려지기 때문에 Bottleneck 구조 사용
- $F$  function에 3개의 layer 사용
- $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$  형식의 layer 사용
- 2-layer의 input/output dimension :  $3 \times 3$
- 3-layer의 input/output dimension :  $1 \times 1$

# Experiments

## Bottleneck Architectures

Model		Performance (FLOPs)
ResNet	18-layers	1.8 billion
	34-layers	3.6 billion
	<b>50-layers</b>	<b>3.8 billion</b>
	<b>101-layers</b>	<b>7.6 billion</b>
	<b>152-layers</b>	<b>11.3 billion</b>
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	<b>21.43</b>	<b>5.71</b>

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43 <sup>†</sup>
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	<b>19.38</b>	<b>4.49</b>

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

# 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	<b>6.43</b> (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	<b>76.4</b>	<b>73.8</b>

Object Detection on PASCAL VOC

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

Object Detection on COCO



TRAIN AND TEST