

ResNet



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Deep Residual Learning for Image Recognition

Kaiming He

Xiangyu Zhang

Shaoqing Ren

Jian Sun

Microsoft Research

{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



Kaiming He

Research Scientist, Facebook AI Research (FAIR)

fb.com의 이메일 확인됨 - [홈페이지](#)

[Computer Vision](#) [Machine Learning](#)

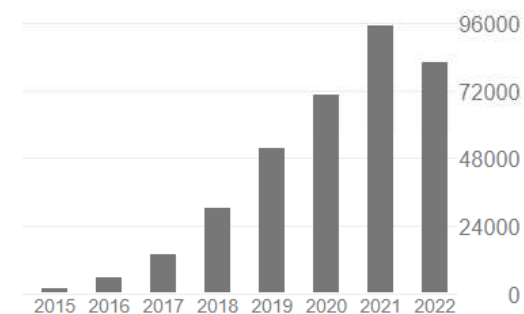
[팔로우](#)

[내 프로필 만들기](#)

제목	인용	연도
Deep Residual Learning for Image Recognition K He, X Zhang, S Ren, J Sun Computer Vision and Pattern Recognition (CVPR), 2016	139721	2016
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks S Ren, K He, R Girshick, J Sun Neural Information Processing Systems (NIPS), 2015	48857	2015
Mask R-CNN K He, G Gkioxari, P Dollár, R Girshick International Conference on Computer Vision (ICCV), 2017	22403	2017
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification K He, X Zhang, S Ren, J Sun International Conference on Computer Vision (ICCV), 2015	17242	2015
Focal Loss for Dense Object Detection TY Lin, P Goyal, R Girshick, K He, P Dollár International Conference on Computer Vision (ICCV), 2017	16341	2017
Feature Pyramid Networks for Object Detection TY Lin, P Dollár, R Girshick, K He, B Hariharan, S Belongie Computer Vision and Pattern Recognition (CVPR), 2017	14829	2017
Learning a Deep Convolutional Network for Image Super-Resolution C Dong, CC Loy, K He, Y Tang	10159 *	2014

인용

	전체	2017년 이후
서지정보	356339	343557
h-index	63	61
i10-index	71	71



공개 액세스

[모두 보기](#)

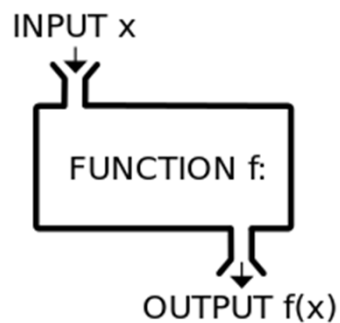
자료 0개

자료 11개

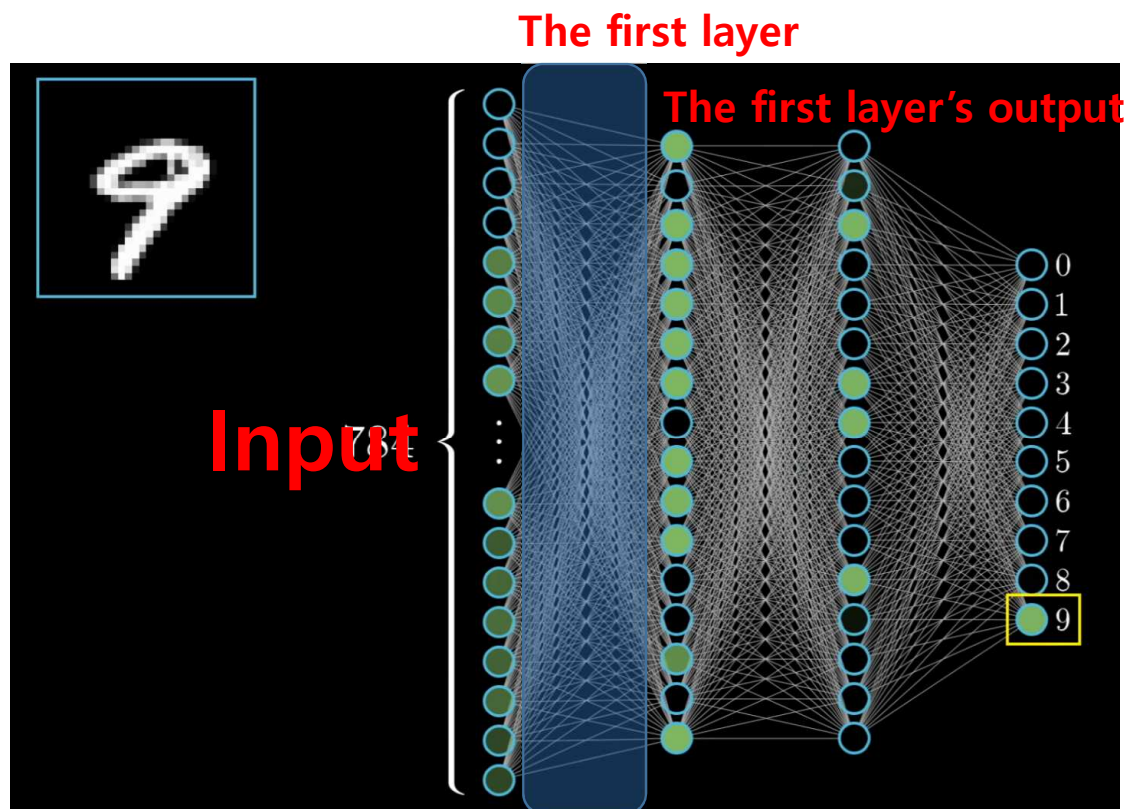
비공개

공개

재정 지원 요구사항 기준



A layer in a deep learning model is a structure or network topology in the architecture of the model, which take information from the previous layers and then pass information to the next layer.



Layer is a sort of function. It gets input and print out output.

The heart of the paper

Observed a problem : Degradation Problem.

- When layers be stacked extremely deep, Accuracy decreased.

The solution this paper suggests : ResNet

- When **ResNet** stacks extremely **deep layers**, it learns well.
- Even **extremely deep ResNet** are easy to optimize.

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—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% 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 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely 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 competitions¹, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

The existing perspectives:

Deepening layers is effective

- The performance of Neural Networks **has improved a lot by deepening layers.**
- As the layer is deepened, **the level of extracted features can be further enriched.**
- **Recent state-of-the-art models** on ImageNet have stacked **16~30 deep layers.**

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 49, 39]. Deep networks naturally integrate low/mid/high-level features [49] and classifiers in an end-to-end multi-layer fashion, and the “levels” of features can be enriched by the number of stacked layers (depth). Recent evidence [40, 43] reveals that network depth is of crucial importance, and the leading results [40, 43, 12, 16] on the challenging ImageNet dataset [35] all exploit “very deep” [40] models, with a depth of sixteen [40] to thirty [16]. Many other non-trivial visual recognition tasks [7, 11, 6, 32, 27] have also greatly benefited from very deep models.

- Asking a question
 - "Really, the more layers the models stack, the better the models learn?"
 - **No. It didn't.** But this problem has been solved recently.
 - Gradient vanishing problem
 - **Somehow, it is solved.** (with Normalized Initialization.)

However..

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 8, 36, 12] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

- **Faced a New Problem**

- "Really, the more layers the models stack, the better the models perform?"

- The problem of gradients vanishing was solved.
The models with extremely deep layers, started to learn anyway.

- **However, the Degradation problem was observed.**

- As the depth of the neural network deepened, the accuracy starts to "saturate".

- **Because the "Training error" increased when network be deepened,
So it was not due to overfitting.**

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [10, 41] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

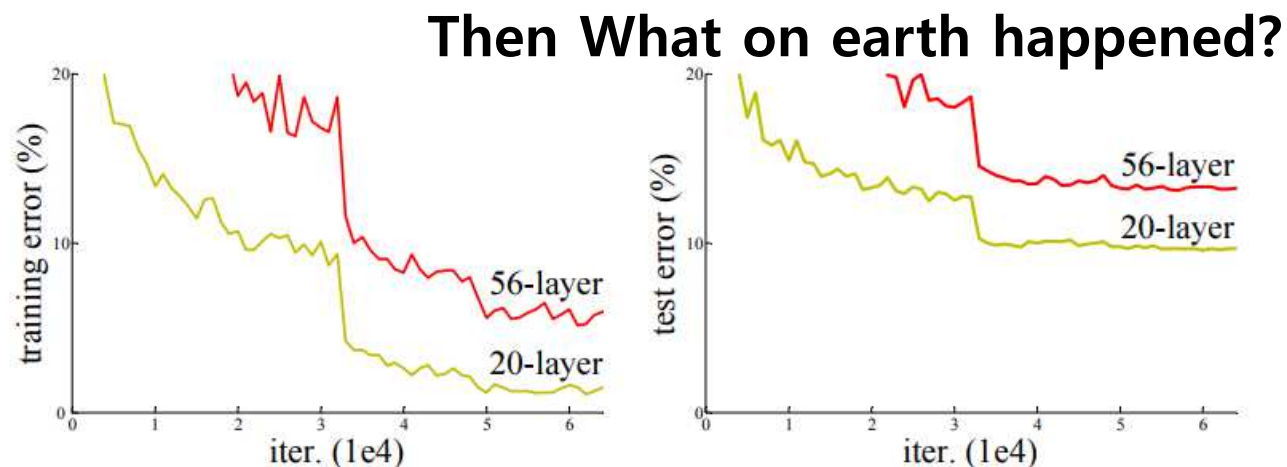


Figure 1. Training error (left) and test error (right) on CIFAR-10

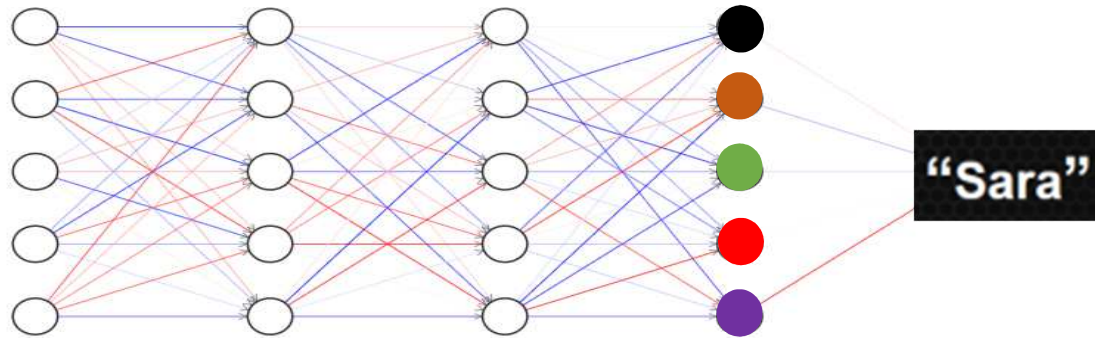
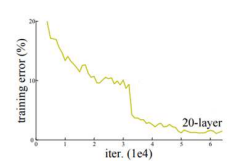


Source : <https://developer.nvidia.com/discover/artificial-neural-network>

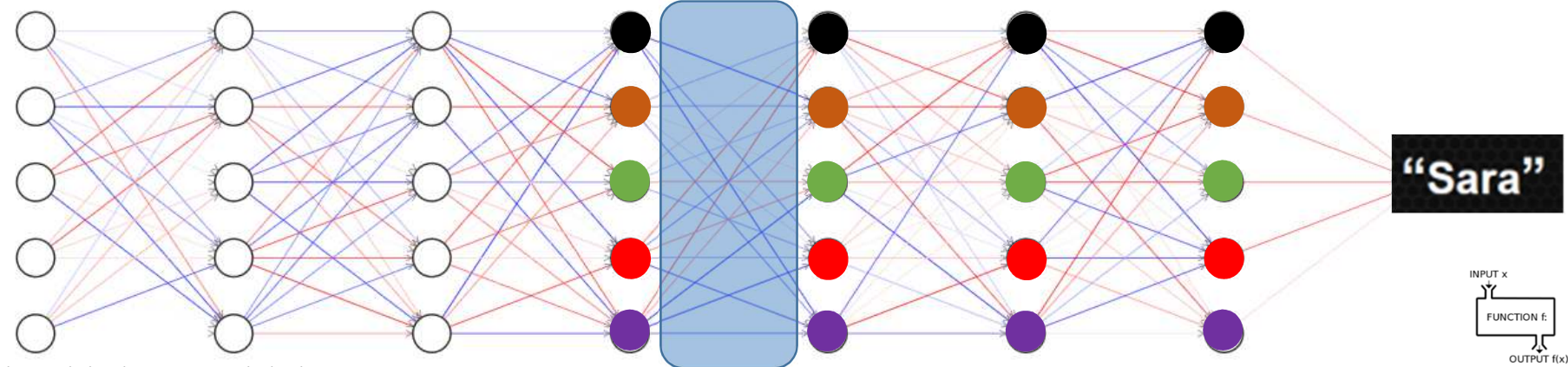
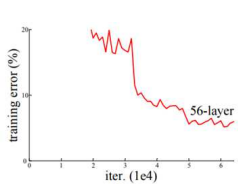
Experiments to explore the problem

The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity mapping*, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution (or unable to do so in feasible time).

Shallower



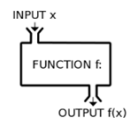
Deeper



Identity mapping : A layer that should approximate identity function.

<https://alexlenail.me/NN-SVG/index.html>

Layer is a sort of function. It gets input and print out output.





Source : <https://developer.nvidia.com/discover/artificial-neural-network>

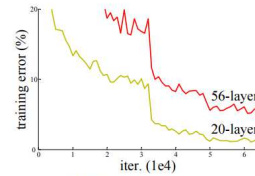


Figure 1. Training error (left) and accuracy (right) vs. iterations (1e4).

Define problem : A problem about the optimization difficulty of deep neural network.

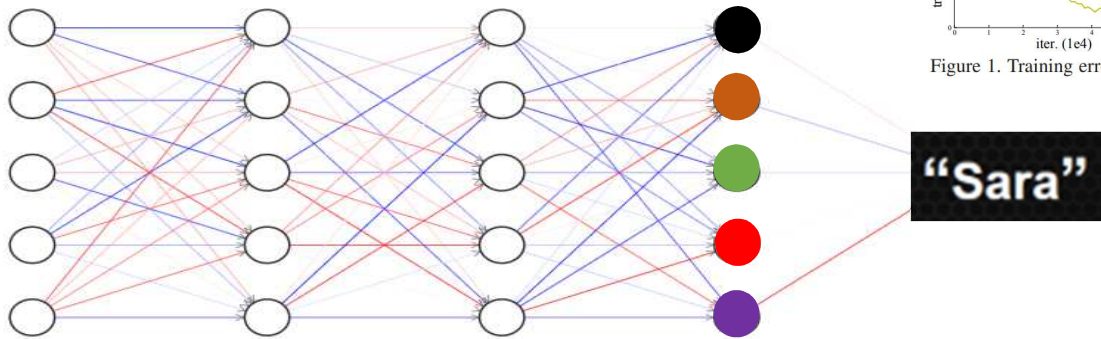
The "Deeper" neural network should perform better, or at least show similar performance than the "Shallower" neural network.

→ It is a fundamental problem about the optimization difficulty of deep neural network.
The deeper, the harder to optimize.

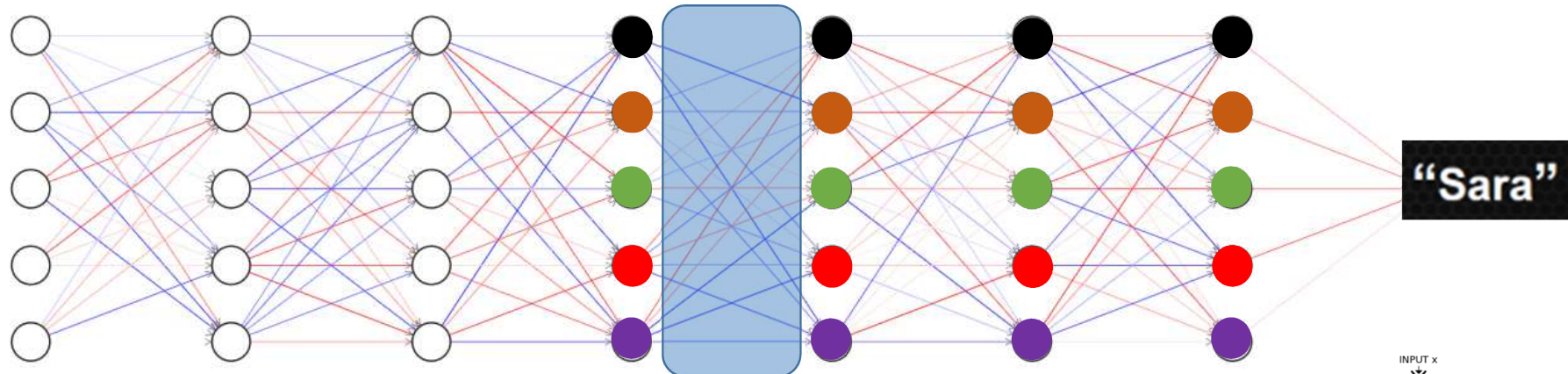
Identity mapping didn't be optimized as a Identity Function

Identity mapping will be closer to Zero Function.
Because, In the first place, the weights flowing through the deep neural network have the property of approaching "0".

Shallower

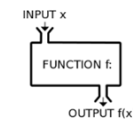


Deeper



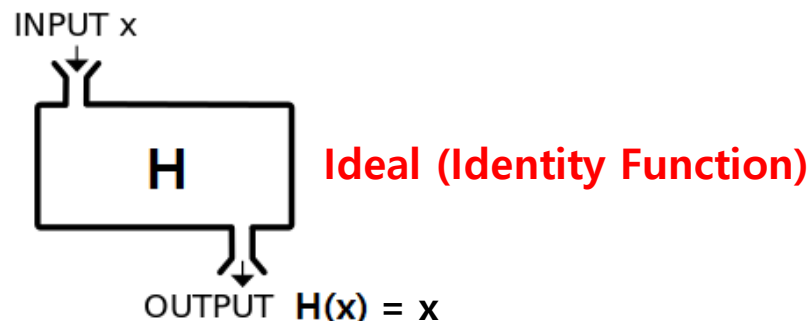
신경망 그림 그려주는 사이트 : <https://alexenail.me/NN-SVG/index.html>

Layer is a sort of function. It gets input and print out output.



Suggests an idea

H is an ideal function that we want to figure out.



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, denoting the desired underlying mapping as $\mathcal{H}(\mathbf{x})$, we let the stacked nonlinear layers fit another mapping of $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original mapping is recast into $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. 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.

But what if Neural Networks could learn $H(x) - x$ instead of H ?

Instead of learning "Ideal" $H(x)$,
The models learn "the gap between ideal and reality", $H(x) - x$, the Residual.

Let's $H(x) - x = F(x)$.

When H should approximate Identity Function :

Just make $F(x)$ "0"

Then $H(x) = 0 + x = x$ and H becomes Identity Function !

c.f. In deep learning, Approximating Zero Function is very easy.

1. Just make all parameters be "0" (whether FCNN or CNN)
2. The weights flowing through the deep neural network have the property of approaching "0". (Xavier Initialization, etc.)

Output : $H(x)$

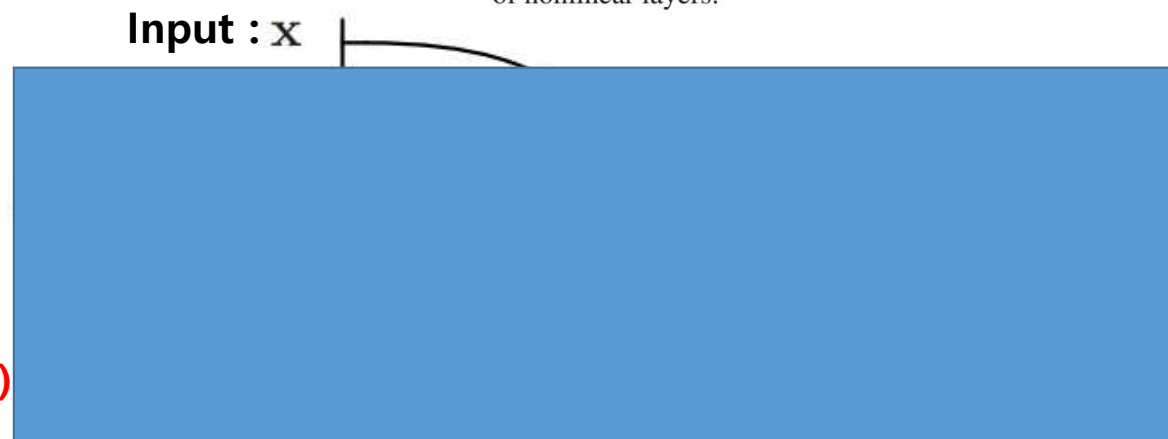


Figure 2. Residual learning: a building block.

Shortcut Connections 설명

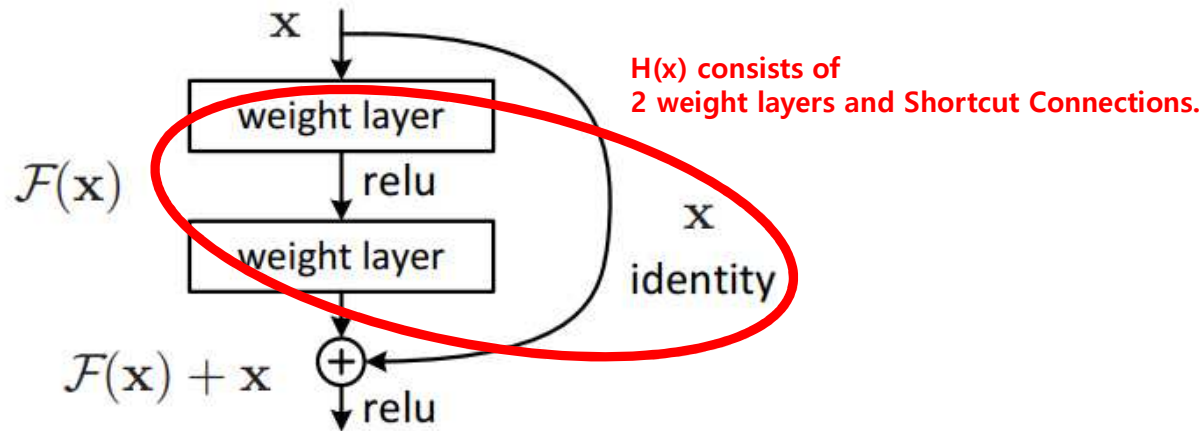


Figure 2. Residual learning: a building block.

Shortcut Connections have no parameters to be learned.
It is just a "command" that add an Input to the output.

Eventually, when updates parameters, only F updates.

Only F learns.

The function of Shortcut Connections

- Without additional parameters, enable to learn Residual Function, F.
- Because there are no additional parameters, plain networks and residual networks can be compared fairly.

The formulation of $\mathcal{F}(x) + x$ can be realized by feedforward neural networks with "shortcut connections" (Fig. 2). Shortcut connections [2, 33, 48] 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.

With this architecture, ResNet is organized and conducted experiments.

- When used ResNet, even extremely deep neural networks were easy to optimize.
 - But, for Plain-net, when networks deepened the training accuracy increased.
- The deeper the network, The higher the accuracy.
 - Recorded the best accuracy among state-of-the-art models.
- Similar results with ImageNet and CIFAR-10.
 - It is not a result for a certain dataset.
- With 152 layers of incredible depth, ResNet is the highest ever, but fewer parameters to learn than VGG net.
 - Thanks to Shortcut Connections
- ResNet has won the following competitions.
 - ImageNet detection in ILSVRC 2015
 - ImageNet localization in ILSVRC 2015
 - COCO detection in COCO 2015
 - COCO segmentation in COCO 2015

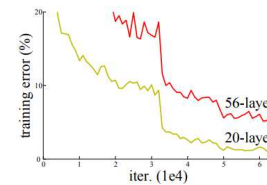


Figure 1. Training error (left) and accuracy (right) versus iteration (1e4) for 20-layer and 56-layer ResNet models.

We present comprehensive experiments on ImageNet [35] 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-10 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 [35], 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 [40]. Our ensemble has **3.57%** 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.

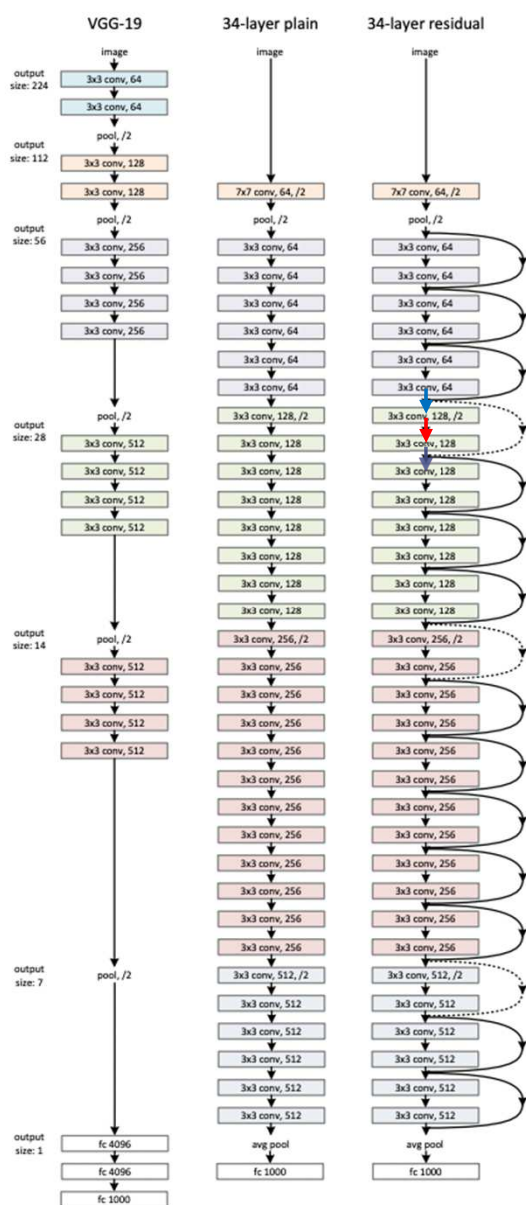
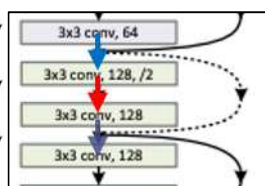


Figure 3. Example network architectures for ImageNet. **Left:** the VGG-19 model [40] (19.6 billion FLOPs) as a reference. **Middle:** a plain network with 34 parameter layers (3.6 billion FLOPs). **Right:** a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Network Architectures

• Plain Network



- I. If the feature map size has been halved, The number of filters should be doubled to preserve the "time complexity" of each layer. (Inspired by VGG philosophy)

The time complexity of an algorithm is the number of basic operations, such as multiplications and summations, that the algorithm performs.

• Residual Network = Plain Network + Shortcuts

- I. Dimension Staying Shortcuts : Identity Shortcuts
- II. Dimension Increasing Shortcuts

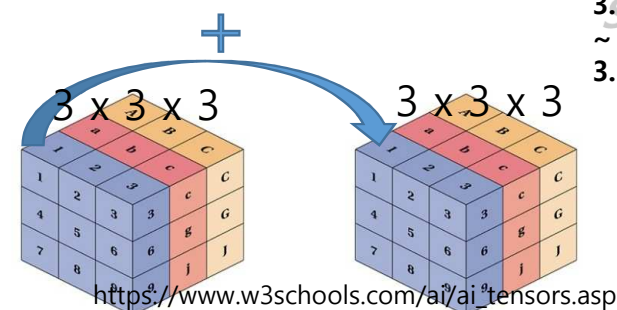
When convolve with stride of 2, Height and Width decreased, Channel numbers increased.

For operations of $F(x) + x$, $F(x)$ and x should have same dimensions.

→ Adjust the dimensions to be identical.

How?

- Zero padding shortcuts
- Projection shortcuts



3.1. Residual Learning

3.3. Network Architecture

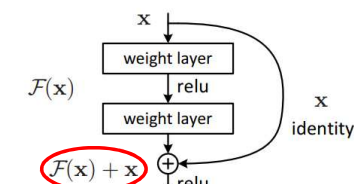


Figure 2. Residual learning: a building block.

4.1. ImageNet Classification

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				
		→ Output dimensions : (56, 56, 64)				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2.

SOURCE CODE FOR TORCHVISION.MODELS.RESNET

```
def conv3x3(in_planes: int, out_planes: int, stride: int = 1, groups: int = 1, dilation:
int = 1) -> nn.Conv2d:
    """3x3 convolution with padding"""
    return nn.Conv2d(
        in_planes,
        out_planes,
        kernel_size=3,
        stride=stride,
        padding=dilation,
        groups=groups,
        bias=False,
        dilation=dilation,
    )

def conv1x1(in_planes: int, out_planes: int, stride: int = 1) -> nn.Conv2d:
    """1x1 convolution"""
    return nn.Conv2d(in_planes, out_planes, kernel_size=1, stride=stride, bias=False)
```

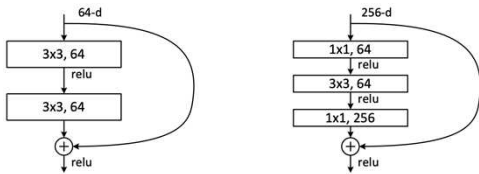


Figure 5. A deeper residual function \mathcal{F} for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

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				
		Zero padding 3 times and convolve				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	Zero padding once and pooling				
		$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

Evaluated on **"ImageNet 2012 Classification Dataset"**

- 1,000 classes
- 128 millions Training set
- 5 millions Validation set

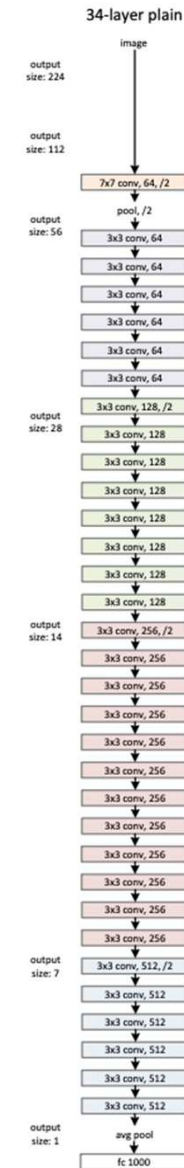
	plain	ResNet
18 layers	27.94	27.88
	↓ Increased	↓ Decreased
34 layers	28.54	25.03

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.

Results :

- the deeper the Plain net, the higher the error
- the deeper the ResNet, the lower the error

layer name	output size	18-layer	34-layer
conv1	112×112		
conv2.x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
conv3.x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
conv4.x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$
conv5.x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$
	1×1		avg
FLOPs		1.8×10^9	3.6×10^9



4.1. ImageNet Classification

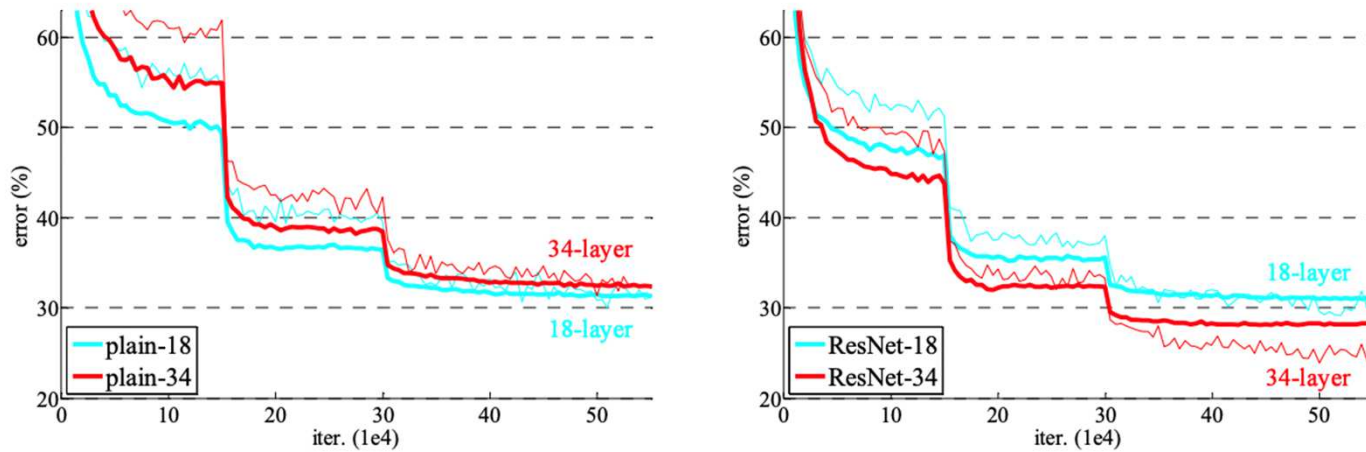


Figure 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

The authors claim that **the error increase** when deepening the Plain net **is not due to gradients vanishing**.

- Already applied methods which have solved gradients vanishing.
- If it is due to gradients vanishing, learning should not have been achieved.
 - But as you see, Plain-34 has competitive accuracy.
 - It means, learning is being achieved to some degree.

4.1. ImageNet Classification

	plain	ResNet
18 layers	27.94	27.88
	↓ Increased	↓ Decreased
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.

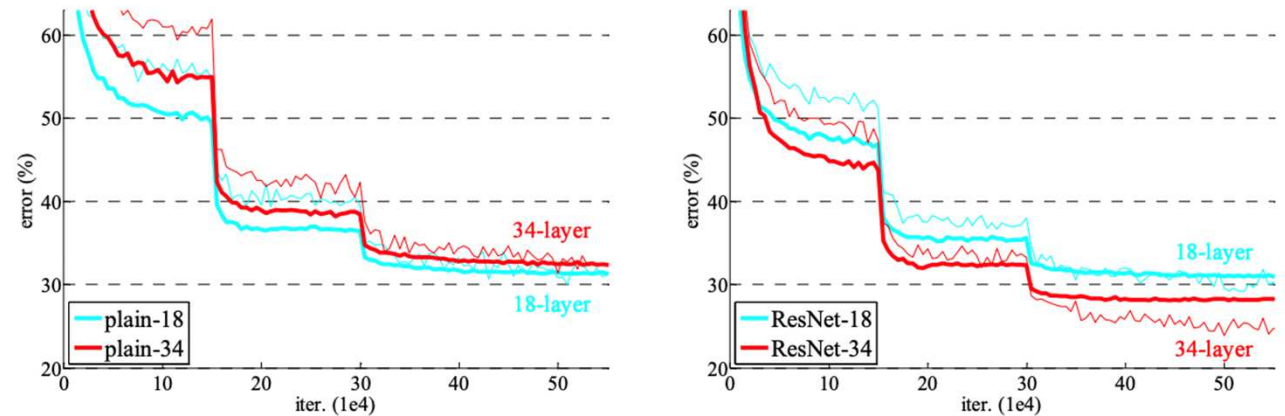


Figure 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Summarize Table 2, Figure 4 :

- **Degradation problem** can be solved with **Residual Learning**.
 - When ResNet be deepened, Training error and Validation error decreased.
- **Verified effects of Residual Learning in Deep Neural Networks.**
 - ResNet-34 decreased error 3.5% compared to Plain-34
- **ResNet optimize much faster than Plain net.**
 - If Neural Network is not very deep, Plain net performs quite good. (27.94 vs 27.88)
 - But you should still use **ResNet. It optimizes very fast.**

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	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.

Option A

Dimension **Increasing** Shortcuts : **Zero padding** shortcuts

Dimension **Staying** Shortcuts : **Identity shortcuts**

→ All shortcuts have no parameters to be learned.

Option B

Dimension **Increasing** Shortcuts : **projection shortcuts**

Dimension **Staying** Shortcuts : **Identity shortcuts**

→ Dimension Increasing Shortcuts introduce parameters to be learned

Option C

Dimension **Increasing** Shortcuts : **projection shortcuts**

Dimension **Staying** Shortcuts : **projection shortcuts**

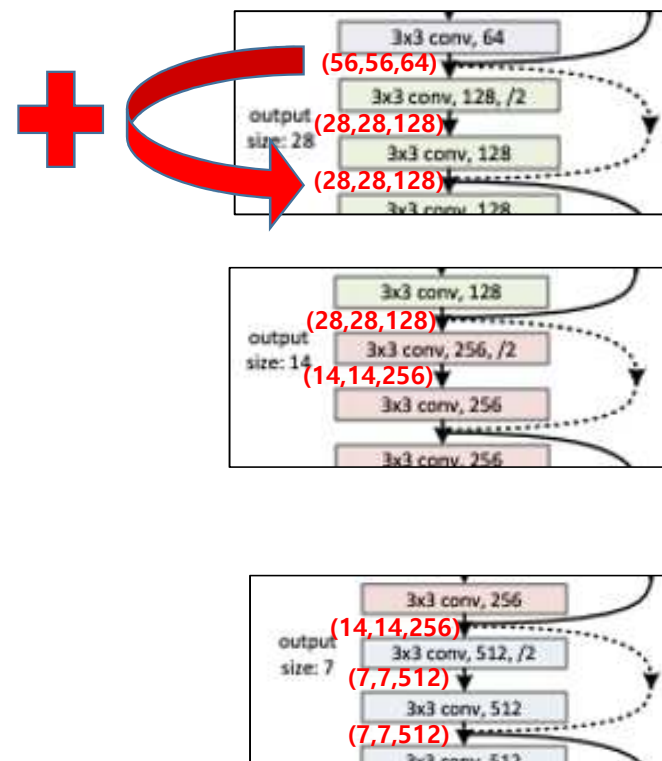
→ → All Shortcuts introduce parameters to be learned

For operations of $F(x) + x$, $F(x)$ and x should have same dimensions.

→ Adjust the dimensions to be identical.

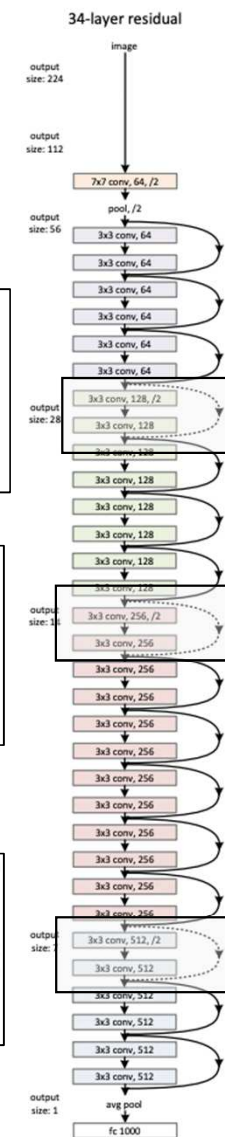
How?

- **Zero padding shortcuts** : **Add channels which have "0" values.**
 - No parameters to be learned.
- **Projection shortcuts** : **1x1 convolution for doubling filter numbers**
 - There are parameters to be learned.



4.1. ImageNet Classification

Identity vs. Projection Shortcuts



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	21.43	5.71

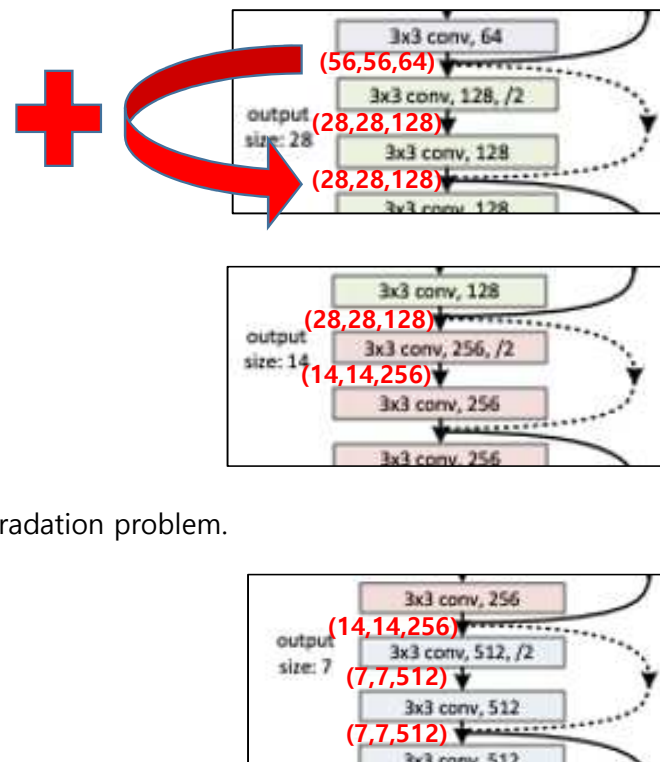
Table 3. Error rates (% , **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

For operations of $F(x) + x$, $F(x)$ and x should have same dimensions.

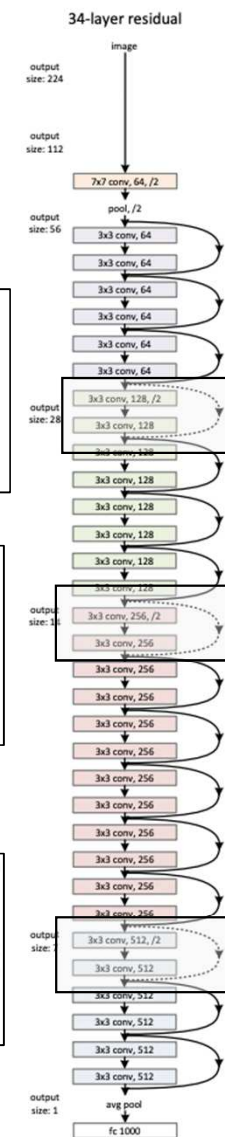
→ Adjust the dimensions to be identical.

How?

- Zero padding shortcuts : **Add channels which have "0" values.**
 - No parameters to be learned.
- Projection shortcuts : **1x1 convolution for doubling filter numbers**
 - There are parameters to be learned.



4.1. ImageNet Classification Identity vs. Projection Shortcuts



summarize the middle part of Table 3

1. With any options, Higher performance than plain-34.
2. With B option is better than with A.
Because, with option A, residual learning isn't achieved in zero padded channels.
3. Insignificant differences between A/B/C → **Projection shortcuts are not essential** to solve degradation problem.
4. Used economic option B on the left experiments.

4.1. ImageNet Classification Deeper Bottleneck Architectures.

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	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.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	$3 \times 3 \times 64 \times 2 \times 3 = 3,456$				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

ResNets with more layers than 34 were also tested.

However, there were too many parameters to use the building block as it was.

So, ResNet-50, ResNet-101, ResNet-152 use new building blocks.

Last FC layer connected to (7,7,2048) increases much FLOPs

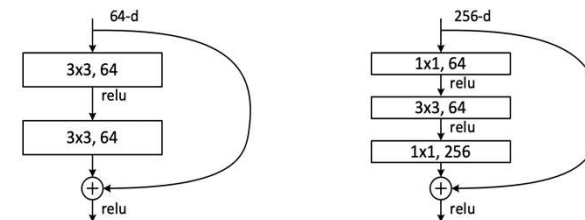


Figure 5. A deeper residual function \mathcal{F} for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

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	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-1 err.	top-5 err.
VGG [40] (ILSVRC' 14)	-	8.43 [†]
GoogLeNet [43] (ILSVRC' 14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	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).

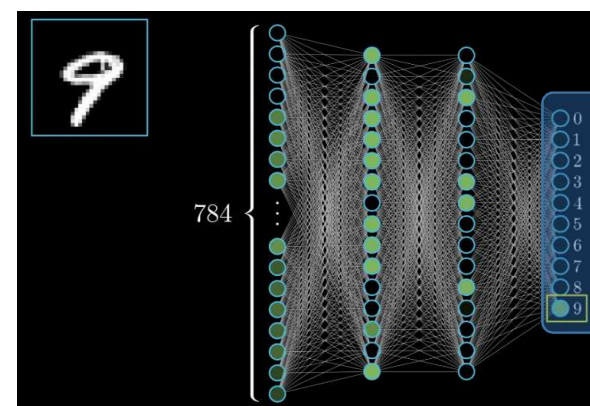
method	top-5 err. (test)
VGG [40] (ILSVRC' 14)	7.32
GoogLeNet [43] (ILSVRC' 14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	3.57

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

결론

1. **ResNet-34 has beaten all the best state-of-the-art deep learning models** (Table 3)
2. **Single model ResNet-152 has beaten all the best state-of-the-art deep learning ensemble models and single models.** (Table 4)
3. **Made a ResNet Ensemble with 6 different ResNets. Won 2015 ILSVRC with Top-5 error 3.57% (Improved existing error 26%.)** (Table 5)


4.1. ImageNet Classification Comparisons with State-of-the-art Methods.



4.2 CIFAR-10 and Analysis

output map size	32×32	16×16	8×8
# layers	$1+2n$	$2n$	$2n$
# filters	16	32	64



Layer name	Output size	20-layer	32-layer	44-layer	56-layer	110-layer	1202-layer
The First layer	(16, 16, 16)	3 X 3, 16, stride 2					
Stacked layers	(16, 16, 16)	3 X 3, 16		conducted experiments with CIFAR-10 Dataset why? In order to focus on analyzing movement of e			
	(16, 16, 16)	3 X 3, 16					
	(8, 8, 32)	3 X 3, 32, stride 2					
	(8, 8, 32)	3 X 3, 32					
	(4, 4, 64)	3 X 3, 64, stride 2					
	(4, 4, 64)	3 X 3, 64					
The last layer	(1,1)	Average pooling, 10-way fully connected layers, softmax					

method			error (%)
Maxout [9]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [34]	19	2.5M	8.39
Highway [41, 42]	19	2.3M	7.54 (7.72±0.16)
Highway [41, 42]	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

Table 6. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [42].

1. ResNet has beaten FitNet and Highway, the existing best Classification models
2. ResNet performs better with much fewer parameters.

Q. By the way, Table 6 shows data augmentation be conducted.

FitNet and Highway was learned with data augmentation, as well ?

→ Reviewing those models' papers, It was. However, augmentation methods differ.

4.2 CIFAR-10 and Analysis

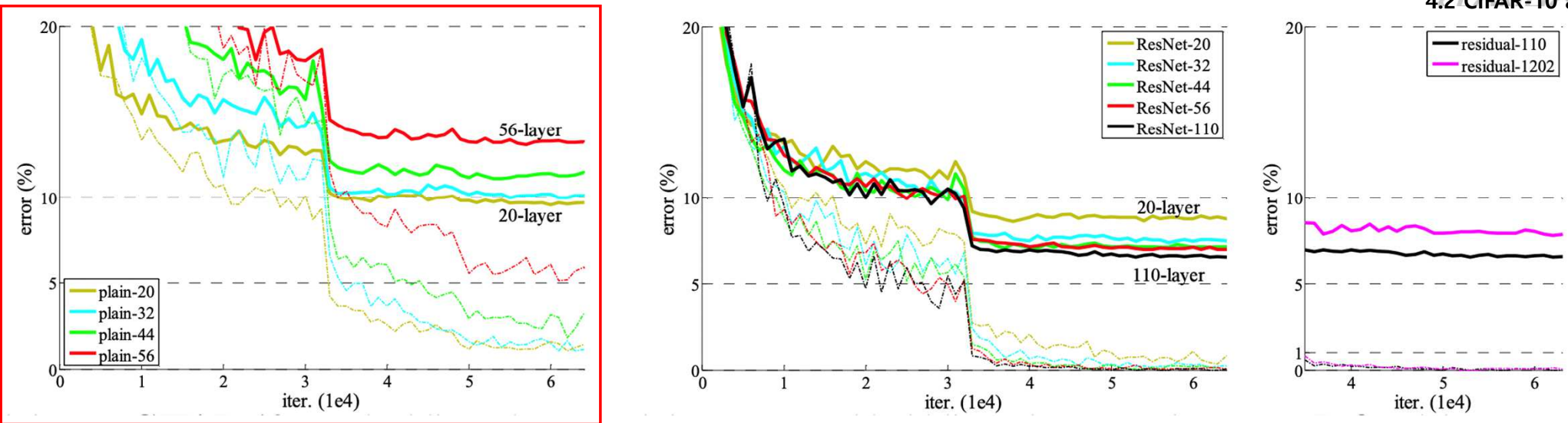
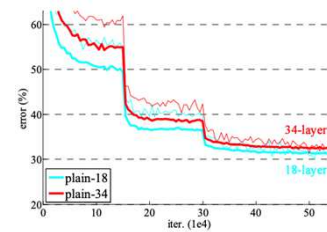


Figure 6. Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error. Left: plain networks. **The error of plain-110 is higher than 60% and not displayed.** Middle: ResNets. Right: ResNets with 110 and 1202 layers.

- the left of Figure 6
The deep plain-net "suffers" from the increased depth, and the training error increases as the depth increases.
- the left of Figure 4 →
For ImageNet Dataset, Similar observations have been made.
- Similar observations have been reported with MNIST dataset. [41]



→ **Conclusion** : *"The optimization difficulty of deep neural networks is a FUNDAMENTAL problem."*
(It's not a matter of overfitting.)

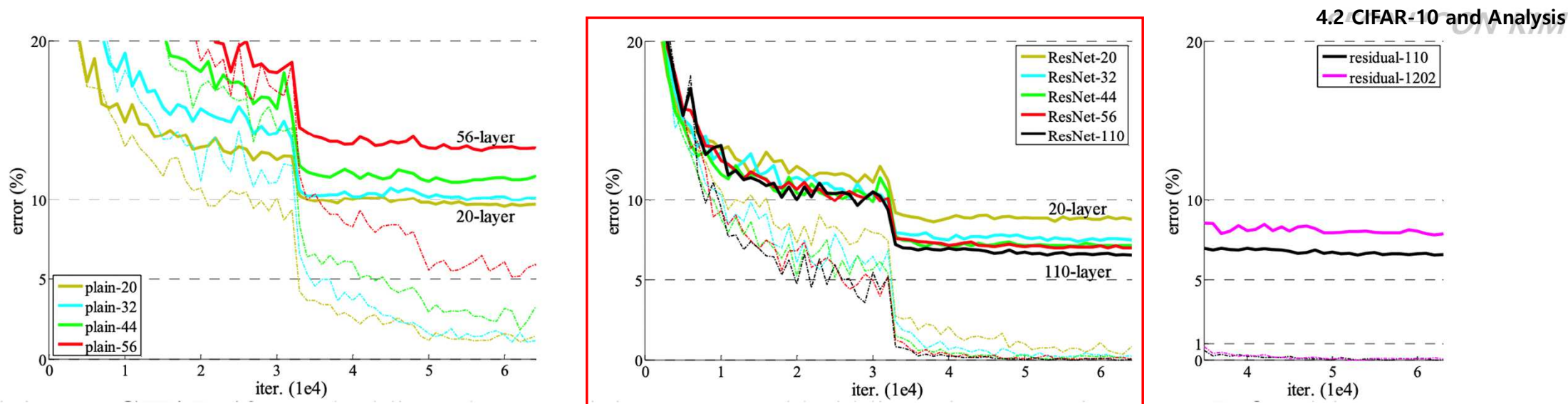
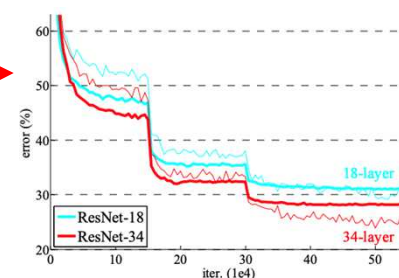


Figure 6. Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error. Left: plain networks. **The error of plain-110 is higher than 60% and not displayed.** Middle: ResNets. Right: ResNets with 110 and 1202 layers.

1. the middle of Figure 6
As depth increased in ResNet, Training error and Testing error decreased.

2. the right of Figure 4 →
For ImageNet Dataset, Similar observations have been made.



→ **Conclusion** : *"ResNet has overcome the optimization difficulty. Accuracy increased when depth increased."*

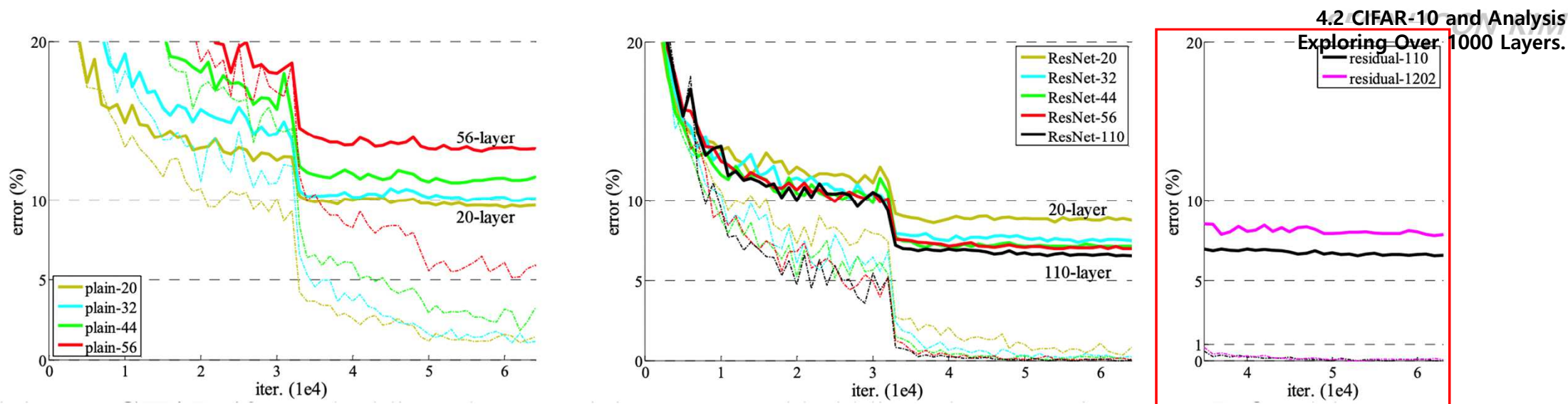


Figure 6. Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error. Left: plain networks. **The error of plain-110 is higher than 60% and not displayed.** Middle: ResNets. Right: ResNets with 110 and 1202 layers.

Thereafter, the ResNet model with 1202 layers was evaluated by setting $n = 200$.

The test error of ResNet-1202 was 7.93%, higher than that of ResNet-110.

→ this may be due to overfitting.

→ Because, the test error is higher although the training error is similar.

The authors speculate that ResNet-1202 would be a "unnecessarily" large model to train small datasets such as CIFAR-10.

4.2 CIFAR-10 and Analysis Analysis of Layer Responses.

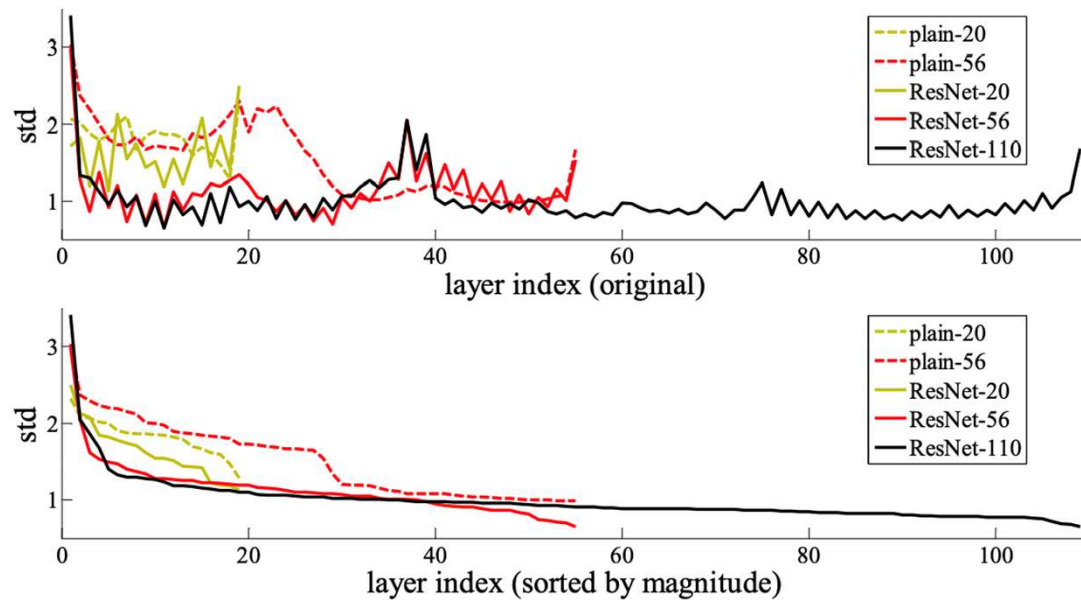


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top:** the layers are shown in their original order. **Bottom:** the responses are ranked in descending order.

Since **ResNet learns residual**, it was expected that there would be **smaller layer responses than Plain-net**.

Figure 7 supports such expectations.

4.3. Object Detection on PASCAL and MS COCO

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 appendix for better results.

VGG-16 is replaced by ResNet-101. Then Performance improved 28%

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 appendix for better results.

- **ResNet has won the following competitions**
 - **ImageNet detection in ILSVRC 2015**
 - **ImageNet localization in ILSVRC 2015**
 - **COCO detection in COCO 2015**
 - **COCO segmentation in COCO 2015**

END