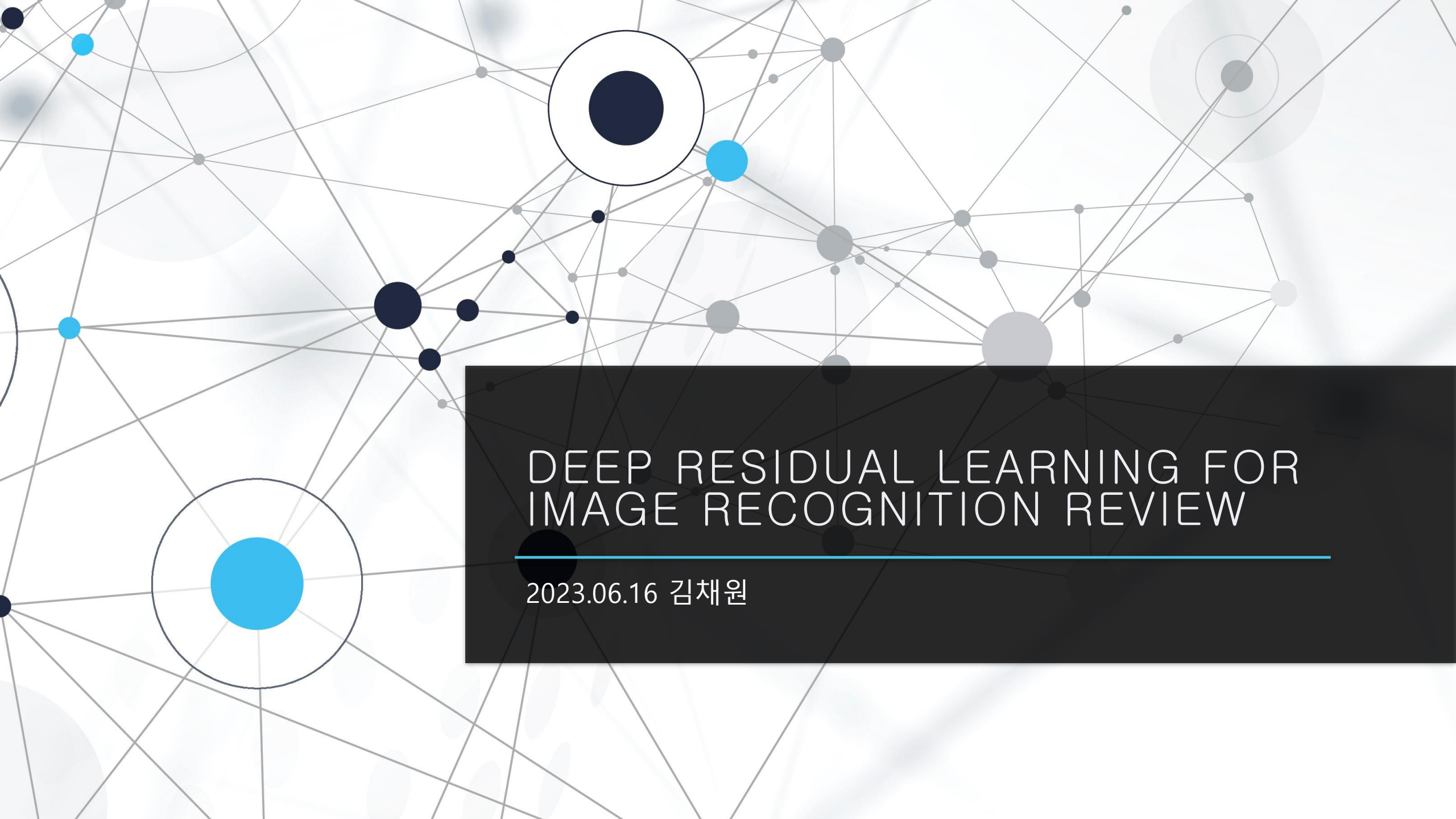
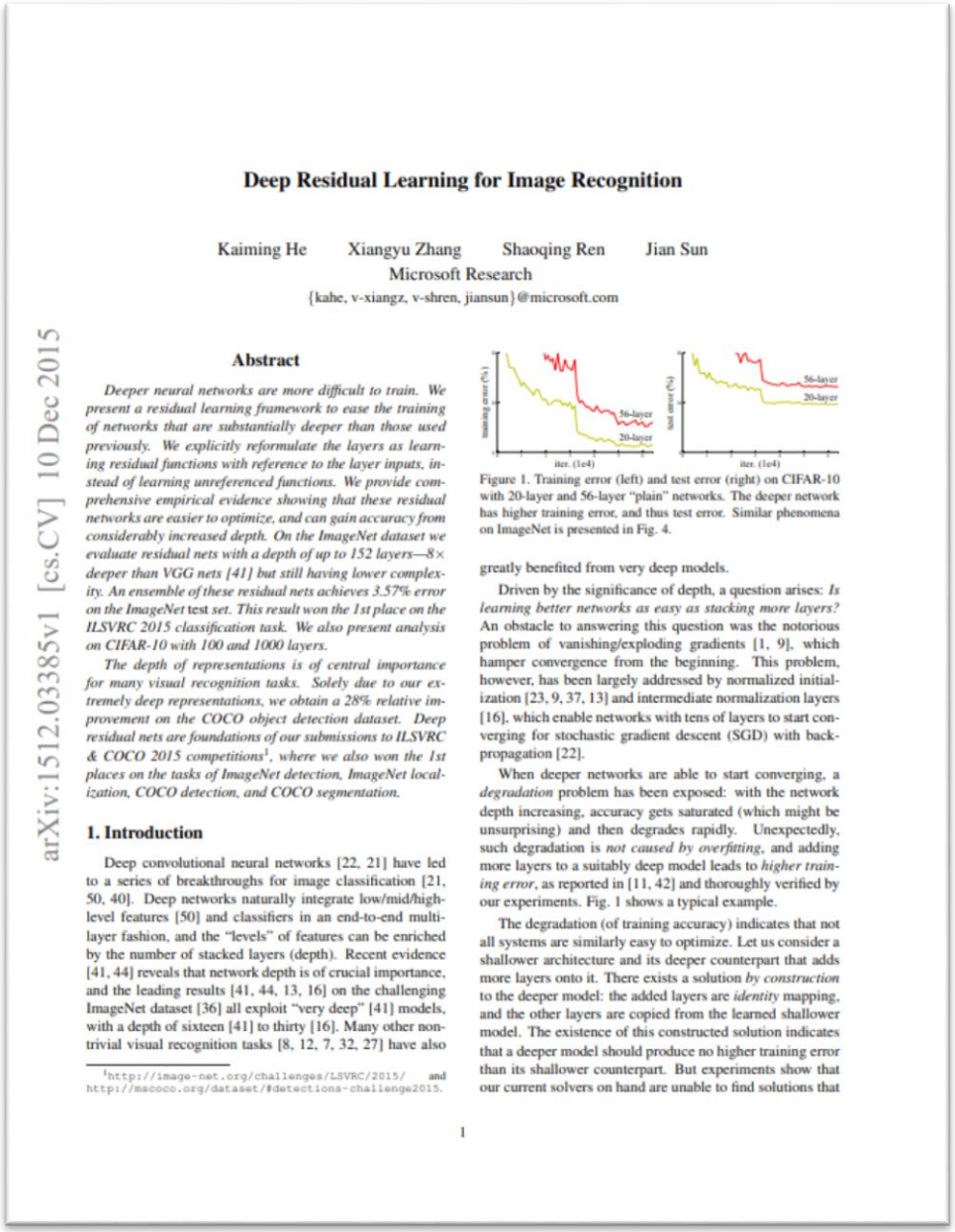


DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION REVIEW

2023.06.16 김채원



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, 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—8× deeper than VGG nets [41] 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 competitions[1], 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, 40]. Deep networks naturally integrate low/mid/high-level features [50] 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 [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also

[1] http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-challenge2015.

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.

greatly benefited from very deep models.

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 [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

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 [11, 42] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

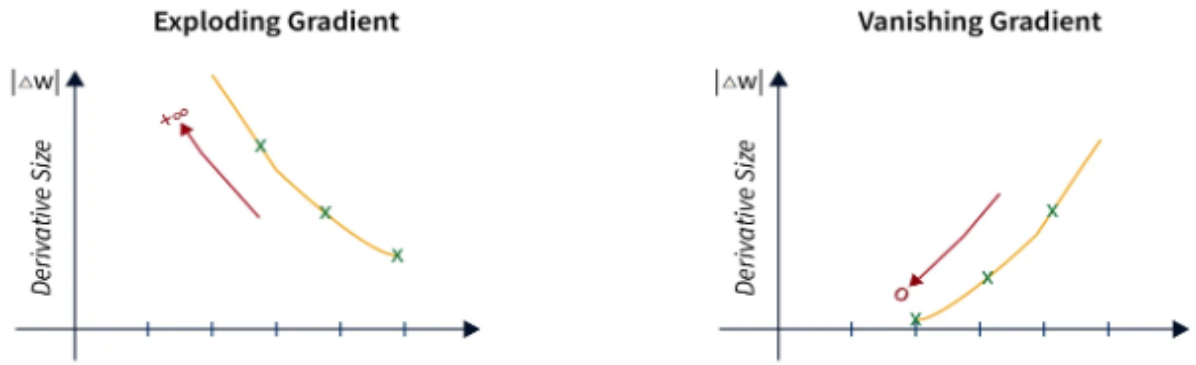
The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, 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

Conference : CVPR (2016)

Published : 2015.12.10

1.Introduction

- *Is learning better networks as easy as stacking more layers?*
- vanishing/exploding gradient



SCALER Topics

1.Introduction

- Degradation Problem

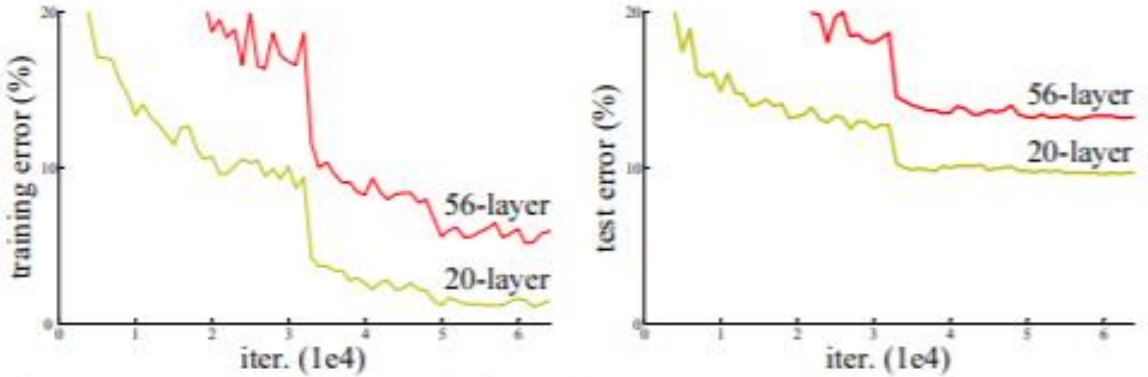


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

1.Introduction

- Deep residual learning

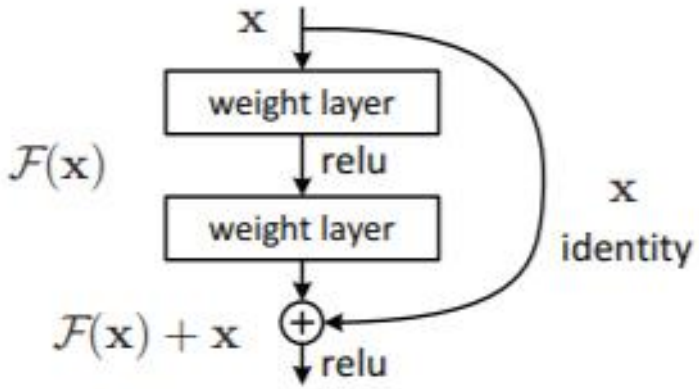


Figure 2. Residual learning: a building block.

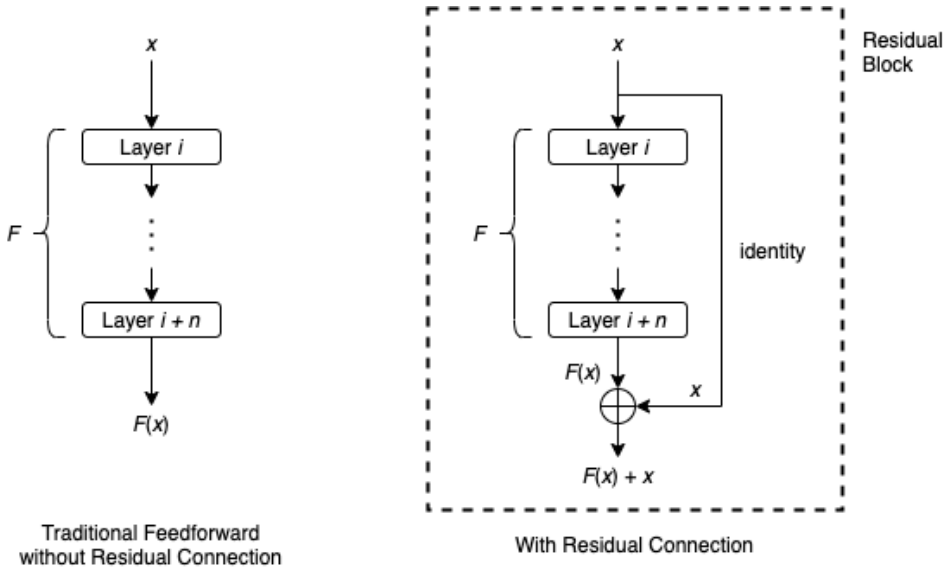
기존 : F(x) = H(x)

Residual learning : F(x) = H(x) – x , 즉 H(x) = F(x) + x

2.Related work

▶ Residual Representations

- In image recognition, VLAD와 Fisher Vector가 좋은 성능을 보임

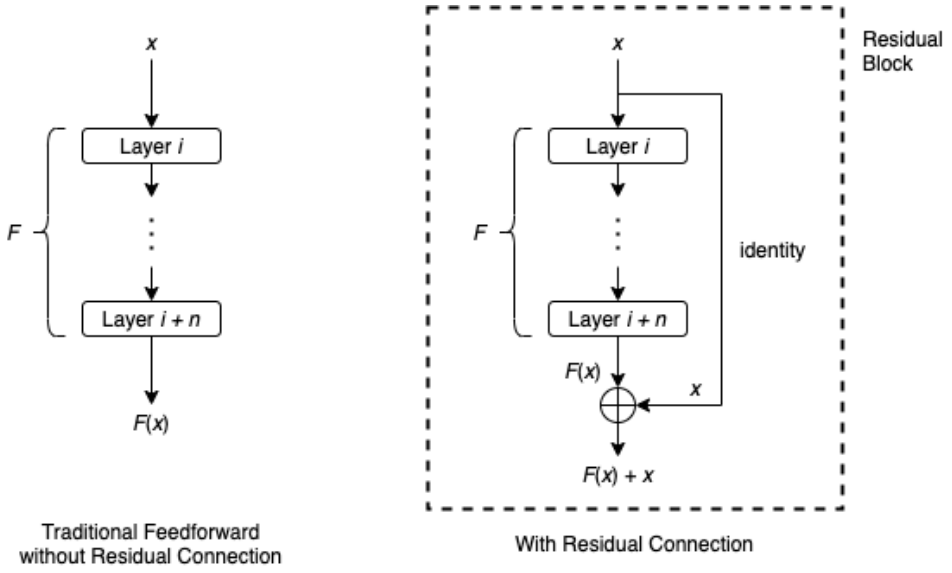


- H(x) = F(x) + x

2.Related work

▶ Shortcut connection(Skip Connection)

- identity mapping으로 shortcut connection이 되게 하면서 x 라는 값을 그대로 전달 (본 논문에서는 별도의 parameter나 complexity를 추가하지 않음)



결국 identity mapping = shortcut connection (값을 그대로 보냄)

2.Related work

▶ 결론

- 레이어를 쌓아서 identity mapping(skip connection)을 맞출 때 기존의 것보다 residual mapping에서 residual을 0으로 만드는 것이 더 쉬움

original	residual mapping
$H(x) = F(x)$	$H(x) = F(x) + x$ $x \rightarrow 0$ Result : $H(x) = F(x)$

- 이 말은 즉, optimizer가 쉬움

3.Method

▶ Residual Learning

- H(x) = F(x) + x
- optimizer가 쉽기때문에 사용자가 weight 값을 더 얻기 쉬움

▶ Identity Mapping by Shortcuts

- 모든 few stacked layer마다 residual mapping

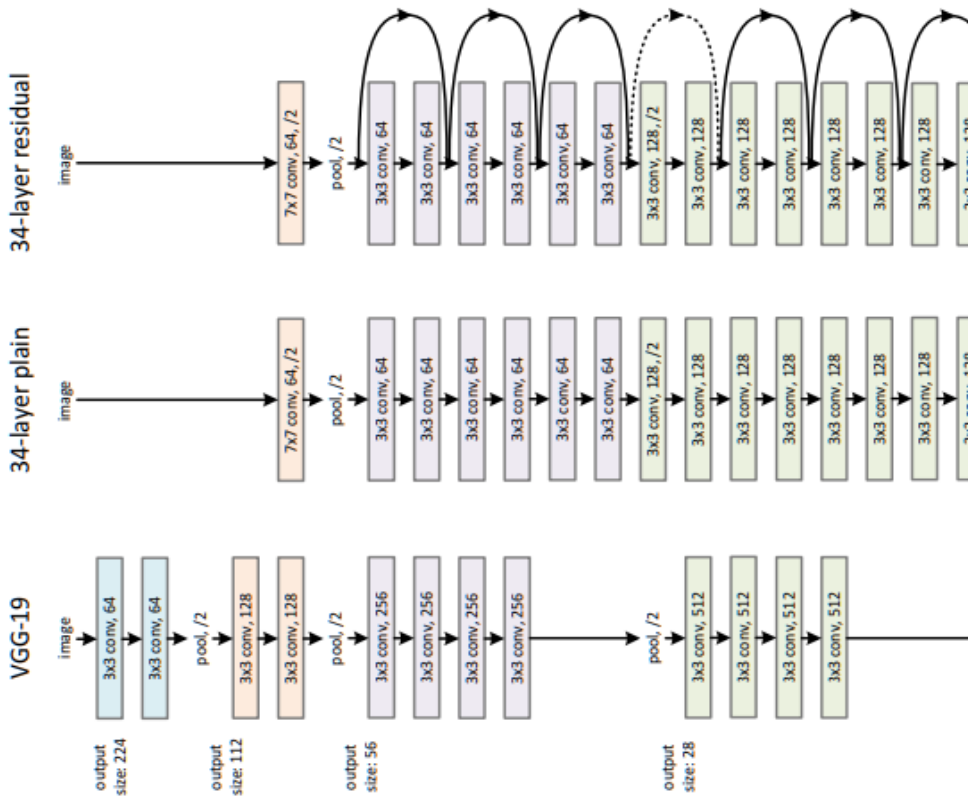
$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \quad (1)$$

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \quad (2)$$

- x와 F의 차원이 같을 것
- 그렇지 않다면, shortcut connection에 W_s 를 적용하여 차원을 맞춤

3.Method

- Network Architectures



3.Method

▶ Network Architectures

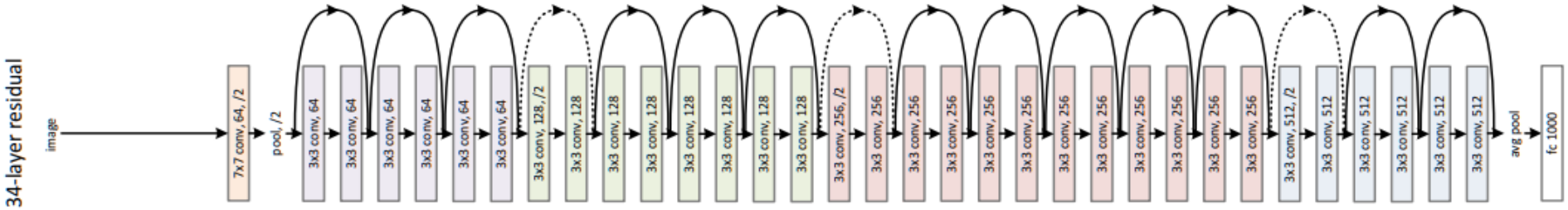
- Plain Network : VGG-19에서 영감 , 최종 레이어 34개
- Convolution layer : 3x3 filter
- filter x 2 -> feature map x ½
- global average polling 사용
- softmax로는 100-way fully-connected layer를 통과시킴

3.Method

▶ Network Architectures

- Residual Network : Plain network + shortcut connection
- x와 F를 같은 차원으로 만들어주기 위해 shortcut 과정에 0을 넣어서 padding (zero-padding)

  또는 $\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$ (projection shortcut)식 사용



4.Results

- ImageNet Classification (vs Plain)

	plain	ResNet
18 layers	27.94	27.88
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.

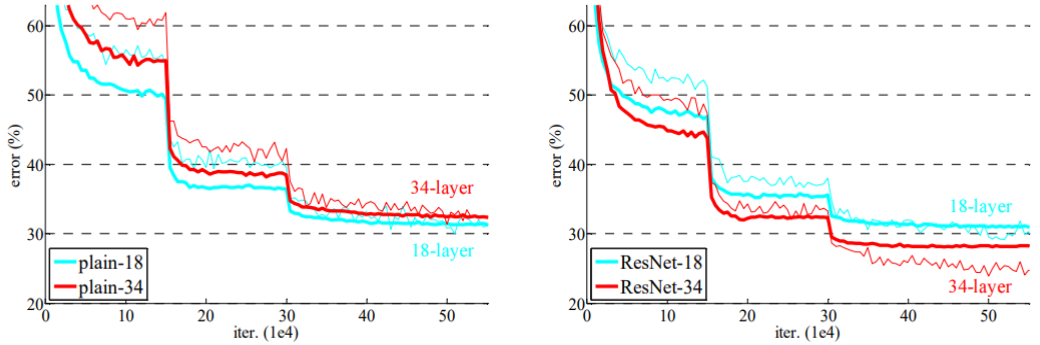


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

- 18-layer ResNet 보다 34-layer ResNet 성능이 더 좋음, layer가 더 깊지만 degradation problem을 잘 조절
- plain과 비교하여도 ResNet이 성능이 더 좋음
- 18-layer 에서 accuracy는 두 모델이 비슷하지만 ResNet모델이 converges가 빨랐음(최적화 향상)

4.Results

▶ ImageNet Classification (vs ResNet)

(A) 증가하는 차원에 대해 zero-padding shortcut, 모든 shortcut은 parameter free

(B) 증가하는 차원에 대해 Projection shortcut(W_s) 적용, 다른 shortcut은 identity

(C) 모든 경우에 Projection shortcut 적용

- 결론: C > B > A

-ABC간의 차이가 작기때문에 projection shortcut이 그렇게 중요한 요소는 아님

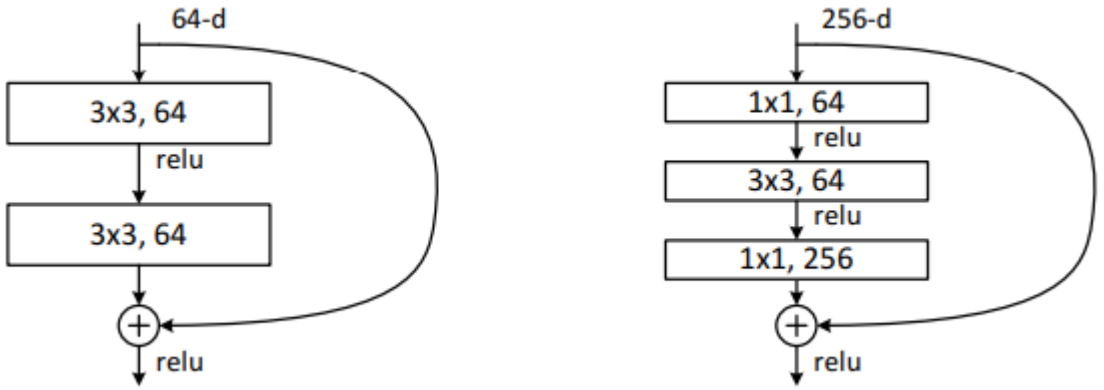
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

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.

4.Results

▶ Deeper Bottleneck Architectures

- Bottleneck design



(1) 50-layer ResNets : 기존의 34-layer에 3-layer bottleneck block 추가

(2) 101-layer and 152-layer ResNets : 더 많은 에 3-layer bottleneck block 추가

4.Results

- Deeper Bottleneck Architectures

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**

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).

결론 : 152 layer까지는 34 layer보다 더 정확함

4.Results

- ImageNet Classification

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.

4.Results

- CIFAR-10 and Analysis
  - 1000개 이상의 layer도 쌓아봤지만 성능이 낮아짐 (overfitting)
  - dropout, maxout 등이 쓰이지 않기 때문에 추후 regularization으로 문제를 해결 해야한다고 말함

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].