Deep Residual Learning for Image Recognition

Katsuya Ogata

May 20, 2025

Overview

Introduction

Very deep networks has made a significant impact on image classification.

- Deep convolutional neural networks have led to a series of breakthroughs for image classification.
- *Is learning better networks as easy as stacking more layers?*
- When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated.

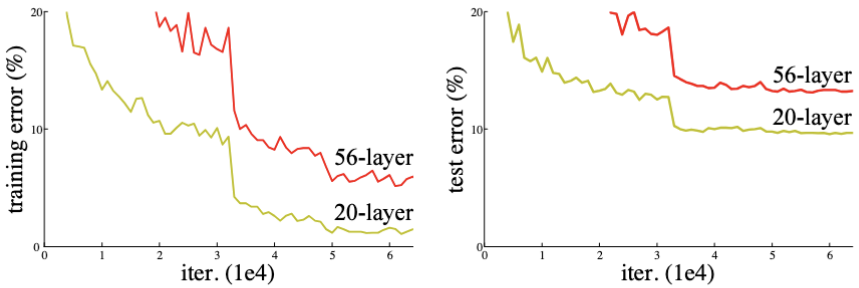


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Introduction

It should be possible for deeper networks to perform at least as well as their shallower counterparts.

- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.
- There exists a solution by construction to the deeper model.
- The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart.
- This paper describes how residual learning and shortcut connections work together to solve the degradation problem.

Related Work

These two ideas (residual representations, shortcut connections) already used in the other works

- Residual Representations.
  - Classic computer vision techniques, such as VLAD and Fisher Vectors, encode “residuals” (i.e., differences from some representative value or dictionary entry) rather than raw features.
- Shortcut Connections.
  - Early MLP variants that included linear connections from input to output use the same idea.
  - The novelty of ResNet lies in using only parameter-free shortcuts (pure identity mapping) and always learning residual functions.

Related Work

The novelty of ResNet lies in using these method in deep networks to optimize training.

- Previous methods have not demonstrated successful training of very deep networks (100+ layers) without optimization issues, especially using only identity shortcuts and residual mappings.
- The authors assert their simple, general approach—stacking residual blocks with identity shortcuts—addresses optimization challenges for deep models and outperforms prior strategies.

3.1. Residual Learning

- We address the degradation problem by introducing a Deep residual learning framework.
- Instead of making layers directly fit a desired mapping $H(x)$, we let them fit a residual mapping:

Residual Mapping

Given $H(x)$, let

$$F(x) := H(x) - x \quad \Rightarrow \quad H(x) = F(x) + x$$

- We hypothesize it's easier to optimize $F(x)$ than $H(x)$ directly.

3.1. Residual Learning

Identity Mapping Case

If the identity mapping is optimal, pushing $F(x) \to 0$ is easier than approximating $H(x) = x$ via nonlinear layers.

- This is implemented with shortcut connections, which skip one or more layers and perform identity mapping.

No Overhead, Easy Integration

Shortcut connections add no extra parameters or computation.
Training can proceed end-to-end using SGD and backpropagation.
Easily implementable with standard libraries (e.g., Caffe).

3.2. Identity Mapping by Shortcuts

$$y = F(x, \{W_i\}) + x \tag{1}$$

- x: Input vector
- y: Output vector
- $F(x, \{W_i\})$: Residual function (e.g., two-layer MLP or CNN)

Example:

$$F = W_2\sigma(W_1 x)$$

where σ is ReLU and biases are omitted.

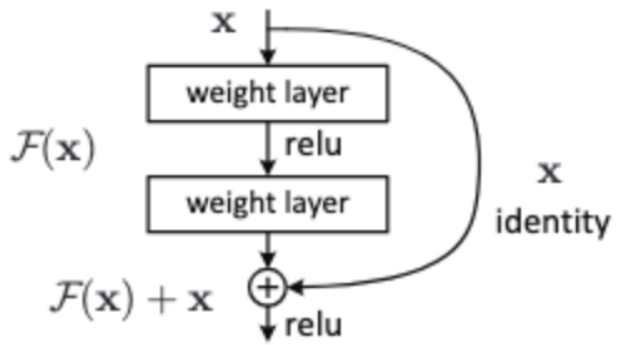


Figure 2. Residual learning: a building block.

3.2. Identity Mapping by Shortcuts When Dimensions Differ:

$$y = F(x, \{W_i\}) + W_s x \tag{2}$$

- W_s: Projection matrix (e.g., 1×1 convolution).
- Used to match dimensions of x and $F(x)$.

Note: Identity mapping ($W_s = I$) is sufficient in most cases.

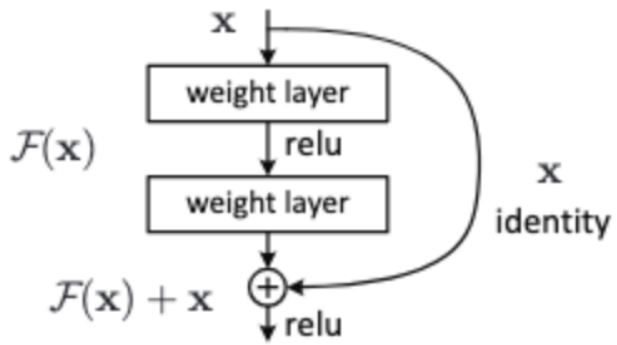
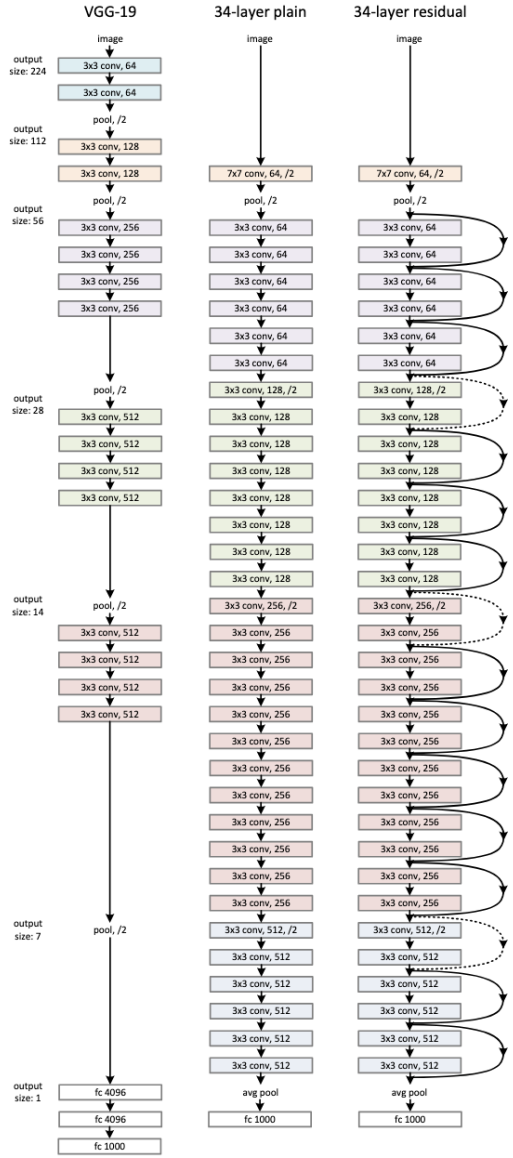


Figure 2. Residual learning: a building block.

Network Architectures

- Based on the plain network, we insert shortcut connections which turn the network into its counterpart residual version.
- When the dimensions increase:
  - (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions.
  - (B) The projection shortcut in Eqn.(2) is used to match dimensions (done by 1×1 convolutions).



Implementation

- 224 × 224 cropping
- color augumentation and horizontal flipping
- adopt batch normalization right after each convolution and before activation
- use He initialization to initialize weights
- the models are trained for up to 60 × 104 iterations
- use a weight decay of 0.0001 and a momentum of 0.9 for SGD
- do not use dropout (*Batch Normalization allows us to use much higher learning rates and be less careful about initialization, and in some cases eliminates the need for Dropout*)

Experiments

4.1. ImageNet Classification

- use datasets that consists of 1000 classes
- 1.28 million training images
- 50k validation images
- 100k test images
- evaluate both top-1 and top-5 error rates

Experiments

We argue that this optimization difficulty is unlikely to be caused by vanishing gradients.

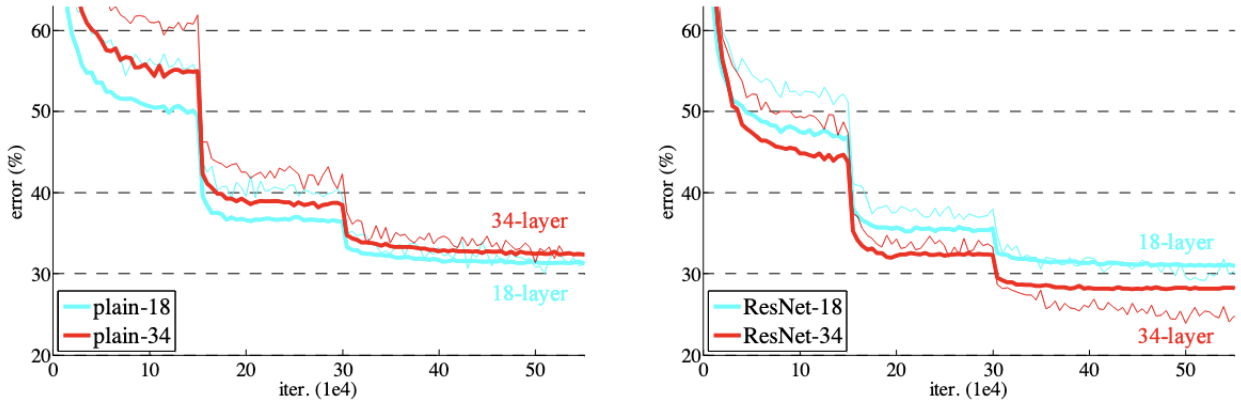


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

	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.

Experiments

the small differences among A/B/C indicate that projection shortcuts are not essential for addressing the degradation problem.

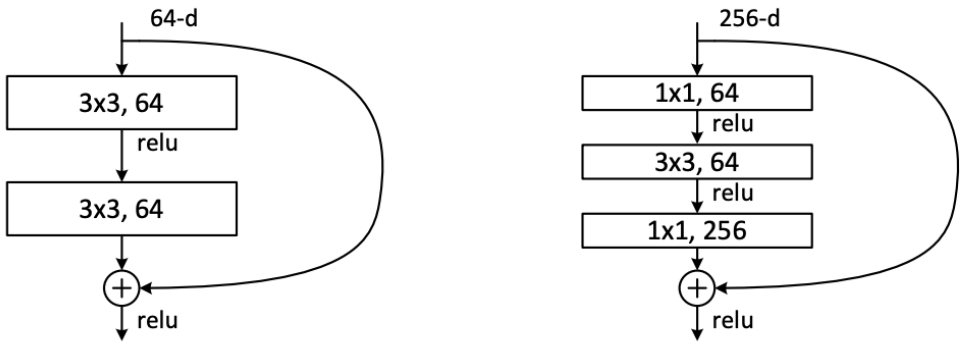
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

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

Experiments

Bottleneck Architecture

- we modify the building block as a bottleneck design
- For each residual function F, we use a stack of 3 layers instead of 2
- The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions.
- The parameter-free identity shortcuts are particularly important for the bottleneck architectures.



Experiments

ResNet showed significant improvements even with deep network.

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**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Experiments

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	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.

4.2. CIFAR-10 and Analysis

- CIFAR-10 dataset, which consists of 50k training images
- 10k testing images
- 10 classes
- Our focus is on the behaviors of extremely deep networks, but not on pushing the state-of-the-art results, so we intentionally use simple architectures
- residual models have exactly the same depth, width, and number of parameters as the plain counterparts

Experiments

It has fewer parameters than other deep and thin networks such as FitNet and Highway.

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

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 [43].

Experiments

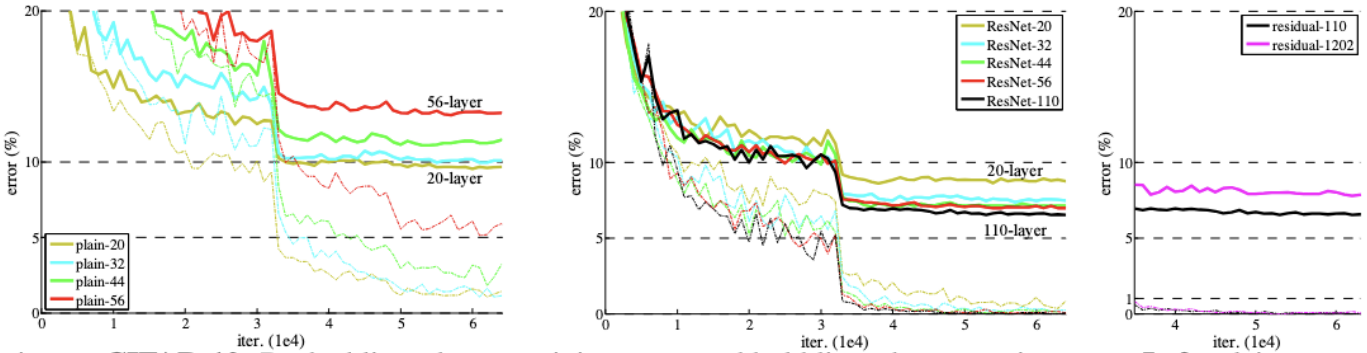


Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

Experiments

ResNets have generally smaller responses than their plain counterparts
The deeper ResNet has smaller magnitudes of responses.

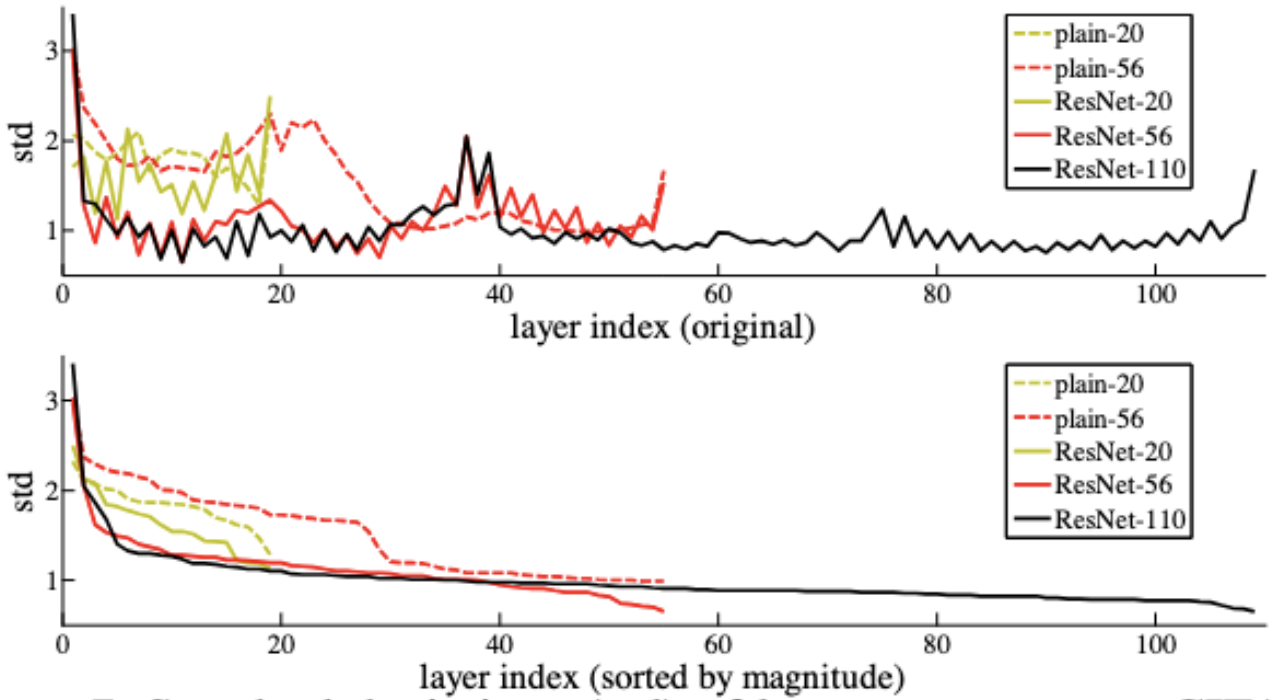


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

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 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.

Appendix

```
Epoch 1, Loss: 0.1566
100%|                                                  | 938/938 [11:04<00:00,  1.41it/s]
Epoch 2, Loss: 0.0405
100%|                                                  | 938/938 [11:03<00:00,  1.41it/s]
Epoch 3, Loss: 0.0352
100%|                                                  | 938/938 [11:03<00:00,  1.41it/s]
Epoch 4, Loss: 0.0278
100%|                                                  | 938/938 [11:02<00:00,  1.41it/s]
Epoch 5, Loss: 0.0261
Test Accuracy: 98.80%
```

Ref:

https://zenn.dev/takeguchi/articles/f6b4530d57177e