DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION

ABSTRACT

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.

INTRODUCTION

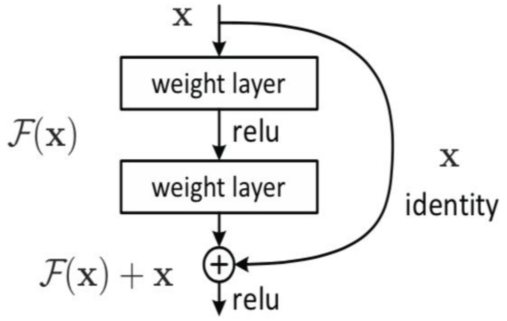
Is learning better networks as easy as stacking more layers?

An obstacle to answering this question was the notorious problem of vanishing/exploding gradients. This problem, however, has been largely addressed by normalized initialization and intermediate normalization layers(BATCHNORM).

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.

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 are comparably good or better than the constructed solution (or unable to do so in feasible time).

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 H(x), we let the stacked nonlinear layers fit another mapping of F(x) := H(x)−x. The original mapping is recast into 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 shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers.Add neither extra parameter nor computational complexity.

The extremely deep representations also have excellent generalization performance on other recognition tasks.

Residual learning principle is generic, and we expect that it is applicable in other vision and non-vision problems.

Related Work.

"highway networks" present shortcut connections with gating functions.

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 formulation always learns residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned.

Residual Learning

If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions , then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, i.e., $H(x) - x$ (assuming that the input and output are of the same dimensions). So <u>rather than expect stacked layers to approximate $H(x)$, we explicitly let these layers approximate a residual function $F(x) := H(x) - x$.</u> The original function thus becomes $F(x)+x$.

The degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings.

It should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one.

...

- $y = F(x, \{W_i\}) + x$
- $F = W_2\sigma(W_1x)$

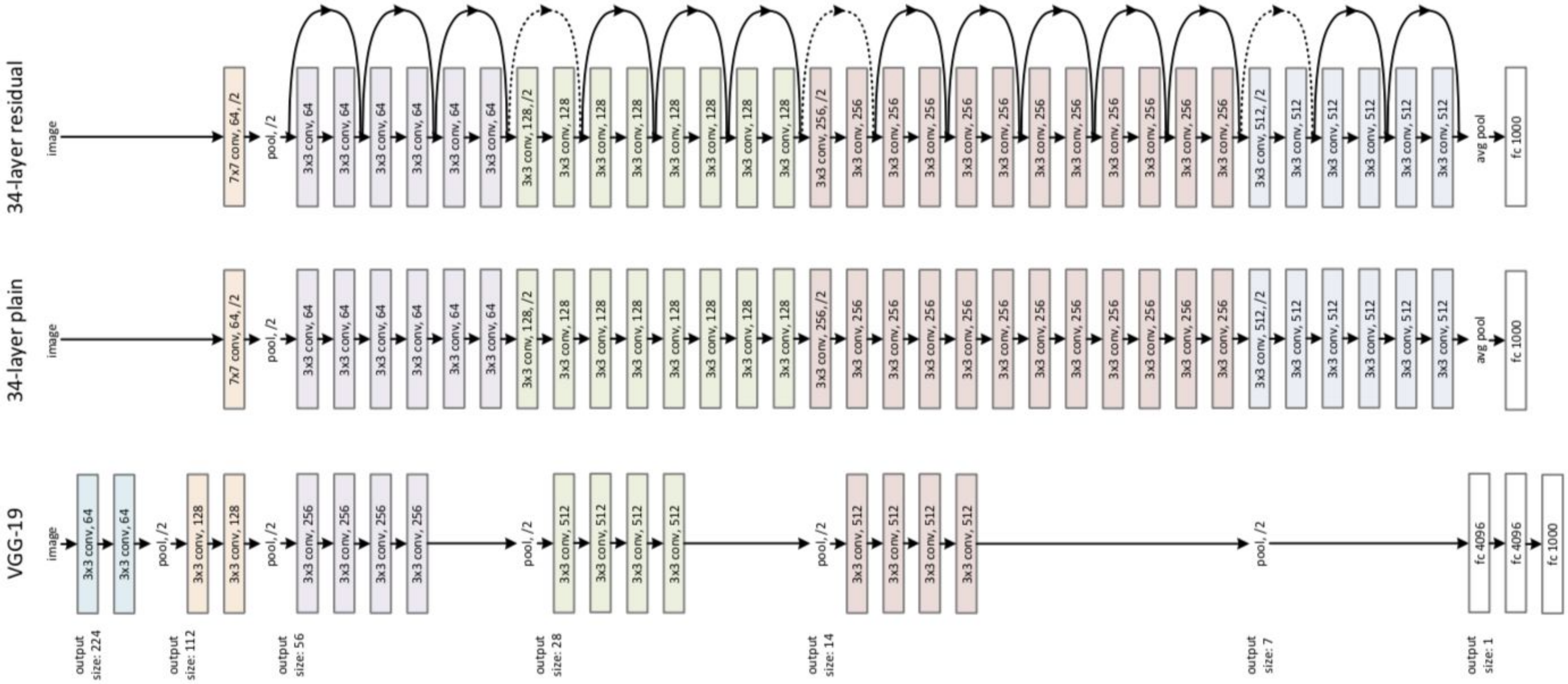
The operation F + x is performed by a shortcut connection and element-wise addition. We adopt the second nonlinearity after the addition.

The dimensions of x and F must be equal in Eqn.(1). If this is not the case (e.g., when changing the input/output channels), we can perform a linear projection W_s by the shortcut connections to match the dimensions.

- $y = F(x, \{W_i\}) + W_sx.$

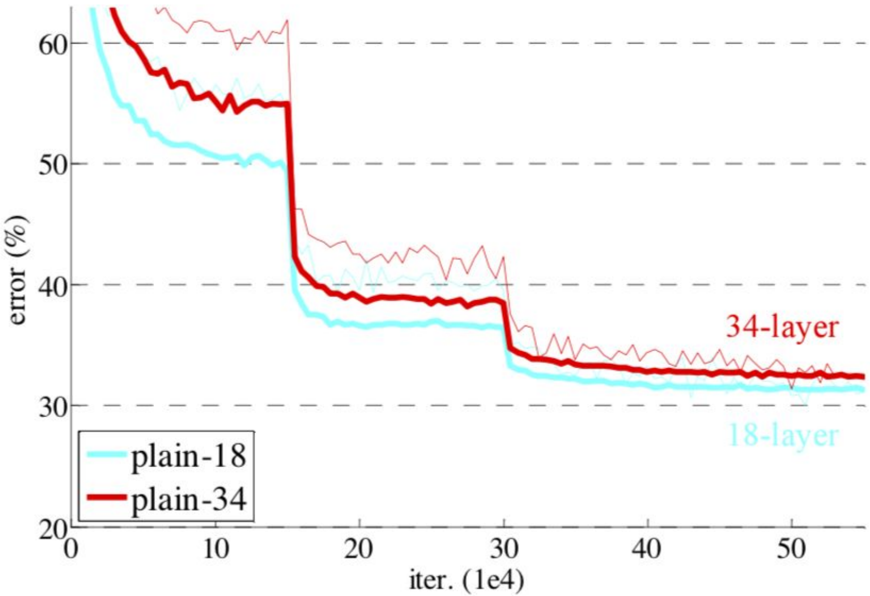
Architecture

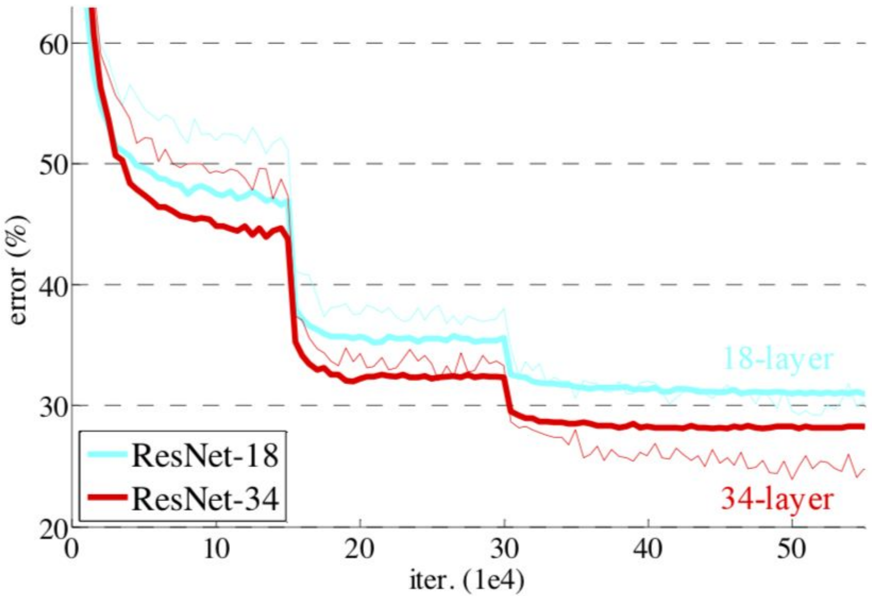
(a) Use padding and identity mapping.
(b) Use projection.



Experiments

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	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


60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34


60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

Types of shortcuts

(A) zero-padding shortcuts are used for increasing dimensions, and all shortcuts are parameter free

(B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity

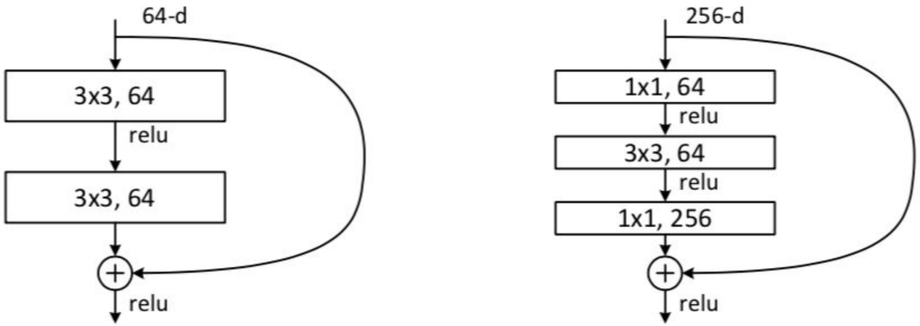
(C) all shortcuts are projections.

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**

B is slightly better than A. We argue that this is because the zero-padded dimensions in A indeed have no residual learning. C is marginally better than B, and we attribute this to the extra parameters introduced by many (thirteen) projection shortcuts. But the small differences among A/B/C indicate that projection shortcuts are not essential for addressing the degradation problem. So <u>we do not use option C in the rest of this paper, to reduce memory/time complexity and model sizes.</u>

Deeper Bottleneck Architecture

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.



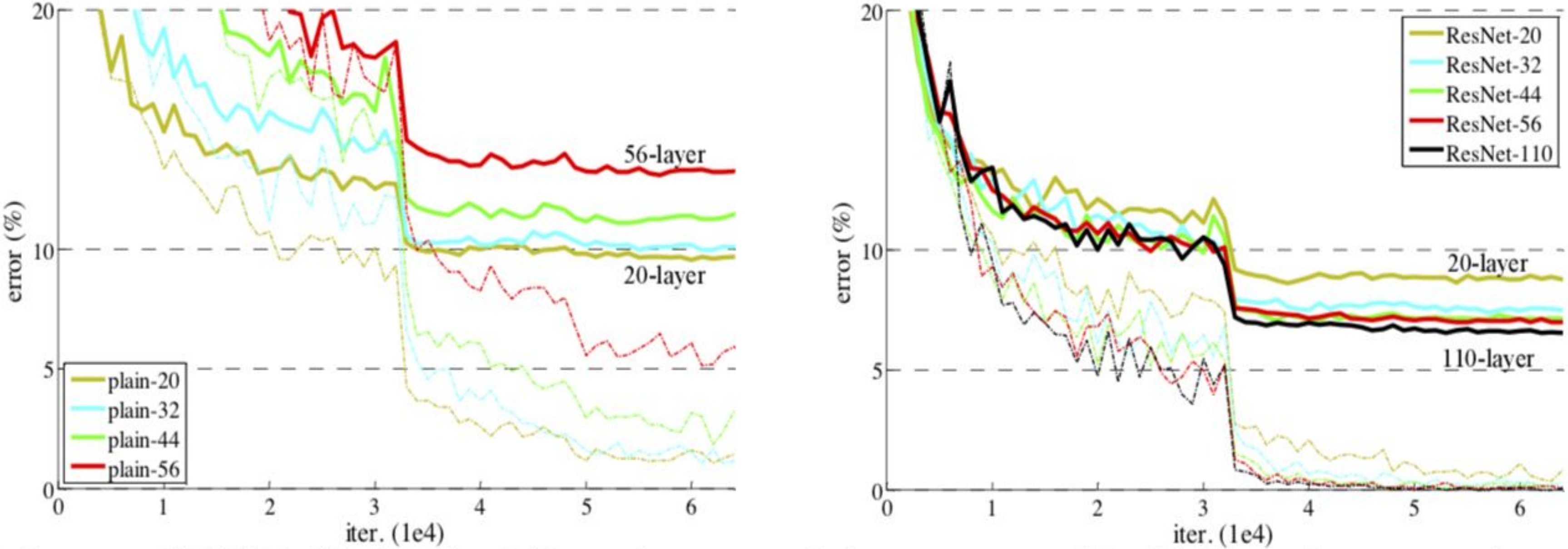
The parameter-free identity shortcuts are particularly important for the bottleneck architectures. If the identity shortcut in is replaced with projection, one can show that the time complexity and model size are doubled, as the shortcut is connected to the two high-dimensional ends. So identity shortcuts lead to more efficient models for the bottleneck designs.

Deeper non-bottleneck ResNets also gain accuracy from increased depth (as shown on CIFAR-10), but are not as economical as the bottleneck ResNets. So the usage of bottleneck designs is mainly due to practical considerations. We further note that the degradation problem of plain nets is also witnessed for the bottleneck designs.

CIFAR-10 And Analysis

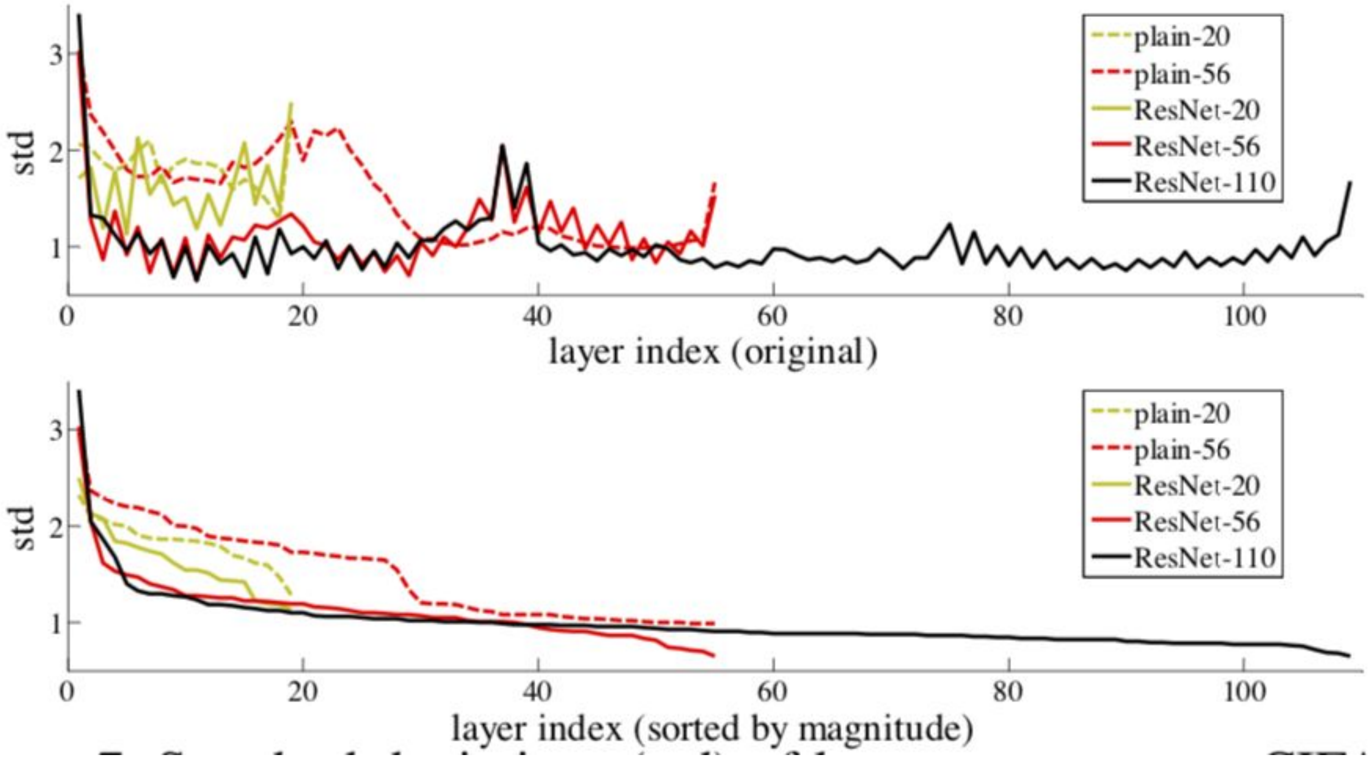
output map size	32×32	16×16	8×8
# layers	$1+2n$	$2n$	$2n$
# filters	16	32	64

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



56-layer
20-layer
error (%)
iter. (1e4)
plain-20
plain-32
plain-44
plain-56
ResNet-20
ResNet-32
ResNet-44
ResNet-56
ResNet-110
20-layer
110-layer
error (%)
iter. (1e4)

Analysis Of Layer Responses



For ResNets, this analysis reveals the response strength of the residual functions. Fig. 7 shows that ResNets have generally smaller responses than their plain counterparts.

When there are more layers, an individual layer of ResNets tends to modify the signal less.

Exploring Over 1000 Layers

There are still open problems on such aggressively deep models. The testing result of this 1202-layer network is worse than that of our 110-layer network.

We argue that this is because of overfitting. The 1202-layer network may be unnecessarily large (19.4M) for this small dataset. Strong regularization such as maxout or dropout is applied to obtain the best results on this dataset .

Object detection improvements on MS COCO using Faster R-CNN and ResNet-101

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

Image Localisation Results

LOC method	LOC network	testing	LOC error on GT CLS	classification network	top-5 LOC error on predicted CLS
VGG's [41]	VGG-16	1-crop	33.1 [41]		
RPN	ResNet-101	1-crop	13.3		
RPN	ResNet-101	dense	11.7		
RPN	ResNet-101	dense		ResNet-101	14.4
RPN+RCNN	ResNet-101	dense		ResNet-101	**10.6**
RPN+RCNN	ensemble	dense		ensemble	**8.9**

method	top-5 localization err	
	val	test
OverFeat [40] (ILSVRC'13)	30.0	29.9
GoogLeNet [44] (ILSVRC'14)	-	26.7
VGG [41] (ILSVRC'14)	26.9	25.3
ours (ILSVRC'15)	**8.9**	**9.0**