Deep Residual Learning for Image Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

https://arxiv.org/abs/1512.03385
https://arxiv.org/pdf/1512.03385.pdf

TL;DR

Deeper neural networks are harder to train; this deep residual learning framework substantially eases training. Layers reference layer inputs instead of learning unreferenced functions. These residual networks are easier to optimize, and continue to gain accuracy from increased depth. Residual nets with more depth are still less complex than VGG nets.

Introduction

Deep convolutional neural networks have been a breakthrough for image classification. Evidence reveals that network depth is of crucial importance. Top-performing architectures benefited from very deep models. However, the phenomenon occurs where degradation happens with more layers. Higher training error, and validation error is present, and is not caused by overfitting. The deep residual learning framework addresses this degradation problem. It explicitly lets stacked layers fit a residual mapping. “Shortcut connections” are added to skip one or more layers. They perform identity mapping, and their outputs are added to the outputs of the stacked layers.

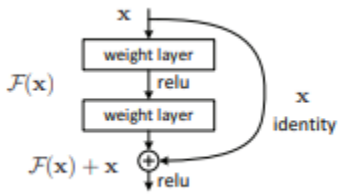
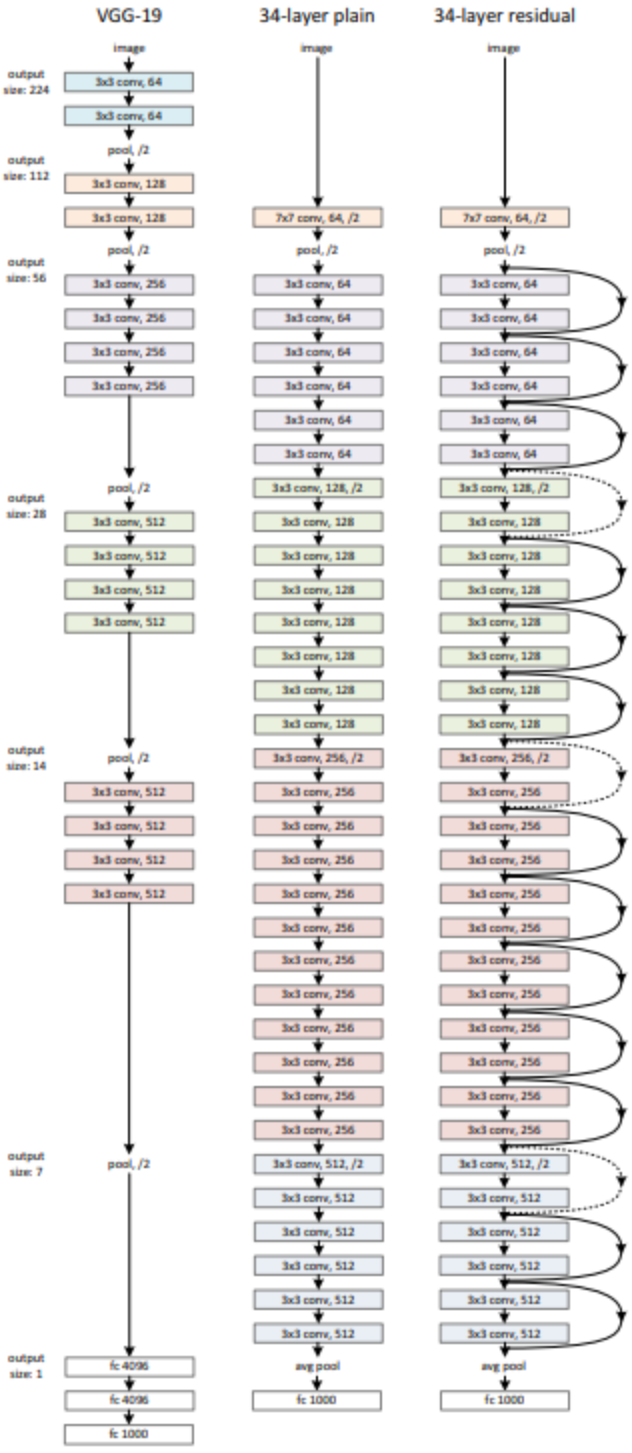


Figure 2. Residual learning: a building block.

Deep Residual learning

Let *H(x)* represent an underlying mapping of a few stacked layers. If some stacked layers can accurately approximate complicated functions, then it is fair to say that it can also approximate a function *H(x) - x*. So rather than expect stacked layers to approximate *H(x)*, we can expect them to approximate a residual function *F(x) = H(x) - x*, or *F(x) + x = H(x)*. Both functions should be able to be approximated, but the training difficulty may be different. This reformulation of the network addresses the degradation problem. The degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. The new framework allows the network to simply drive the weights of a layer towards zero if identity mappings are already ideal (*F(x) + x = H(x)* $\rightarrow$ *0 + x = H(x)*). In real life, it is unlikely that identity mappings are completely optimal, but the reformulation simplest the problem for the solver if the solution for the identity mapping is closer to a zero mapping.



Experiments

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 [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).

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.

The VGG-19 model has 19.6 billion FLOPS, while the plain and residual 34 layer networks have 3.6 billion.